Getting Started Using Universal Mechanism

This manual leads you through the basic possibilities of Universal Mechanism software and shows you how to create and simulate models of several simple mechanical systems. It assumes that you go through the manual step by step sequentially.

Simulation of such mechanical systems as cars and railway vehicles has certainly its own features but basic concepts using UM still the same. These concepts are shown in this manual.

Copyright and trademarks

This manual is prepared for informational use only, may be revised from time to time. No responsibility or liability for any errors that may appear in this document is supposed.

Copyright © 2009 Universal Mechanism Software Lab. All rights reserved.

All trademarks are the property of their respective owners.

Contact information

The latest UM version as well as up-to-date UM user's manual available at http://www.umlab.ru/download.htm.

Please, send your bug reports, questions and suggestions to um@umlab.ru.

Address:

Russia, 241035, Bryansk,
bulv. 50-let Oktyabrya, 7
Bryansk State Technical University,
Laboratory of Computational Mechanics,
Prof. Dmitry Pogorelov

Phone, fax: +7 4832 568637.

1. MODEL OF A PENDULUM 4

1.1. What we will learn 4

1.2. Model scheme 5

1.3. Creating the model 6
1.3.1. Running UM Input and creating new model 6
1.3.2. Familiarizing yourself with Universal Mechanism 6
1.3.3. Creating graphical objects 8
1.3.3.1. Scene image 8
1.3.3.2. Image of pendulum 14
1.3.4. Creating rigid bodies 15
1.3.5. Creating joints 16
1.3.6. Saving the model 17
1.3.7. Preparation for simulation: generation of equations of motion 18
1.3.7.1. Numeric-iterative method 19
1.3.7.2. Symbolic method 20
1.3.8. Run UM Simulation program 21

1.4. Simulation of motion 22

1.5. Multibody pendulum 30
1.5.1. Development of the simple pendulum model 30
1.5.2. Usage of subsystem technique 32
1.5.2.1. Compound object structure 32
1.5.2.2. Subsystem preparation 34
1.5.2.1. Compound model creation 37
1.5.2.1.1. Scene image creation 37
1.5.2.1.2. Subsystems adding 39
1.5.2.1.3. Creating joints 41
1.5.2.2. Simulation 43
1.5.2.3. Connections usage 48

2. FREE AND FORCED OSCILLATIONS 55

2.1. What we will learn 55

2.2. Model scheme 55

2.3. Creating the model 56
2.3.1. Running UM Input and creating new model 56
2.3.2. Creating graphical objects 57
2.3.3. Creating rigid bodies 61
2.3.4. Creating joints 63
2.3.5. Creating force elements 66
2.3.6. Visualization of spring and damper 67
2.3.7. Additional parameters 69
2.3.8. Preparation for simulation 71

2.4. Simulation of motion 72
2.4.1. Free oscillations 72
2.4.2. Statistical analysis 78
2.4.3. Linear analysis 79
2.4.4. Forced oscillations 81

3. SUBSEQUENT STUDYING UNIVERSAL MECHANISM 83

1. Model of a pendulum

1.1. What we will learn

This lesson shows you how to create new model, add rigid bodies and joints, generate and compile equations of motion, simulate dynamics of a model and obtain plots of various performances of the model. This lesson is devoted to general overview of the UM possibilities and workflow.

At the end of the lesson we will have the model of the pendulum (you can find the final model in the **{um_root}\samples\tutorial\eng\pendulum** directory)[1], which will include one rigid body – pendulum itself, one rotational joint and graphical object of the environment – support. After describing the model we will go through the all stages of the working with the model: synthesis and compiling of equations of motion, and then will come to the simulation of motion of the pendulum.

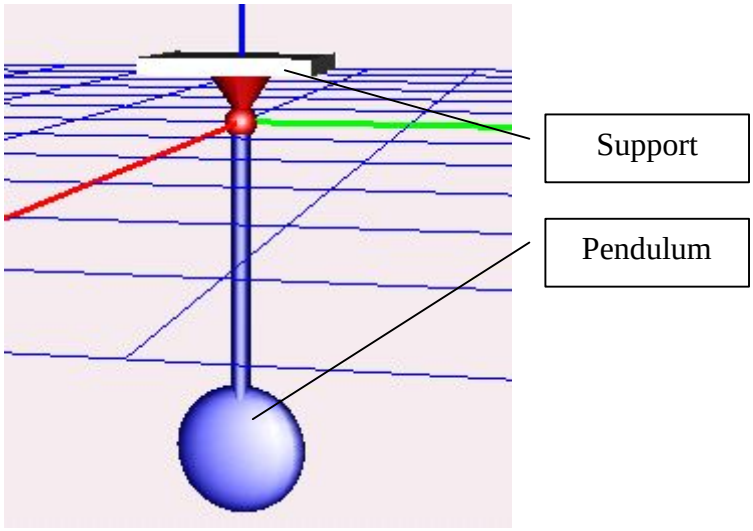
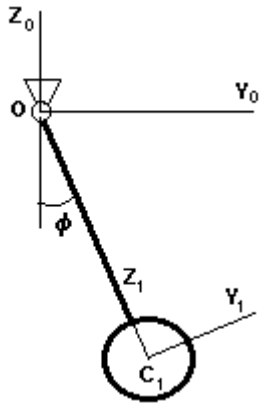


Figure 1. Complete model

Then we will create the model of the multi-link pendulum by the development of the pendulum model and with the usage of the subsystem technique. We will learn how to create compound models, and features of working with mechanical systems with closed kinematical loops.

[1] Pendulum model is also available at http://www.umlab.ru/download/50/eng/pendulum.zip

1.2. Model scheme



Before modeling the pendulum with the help of UM we recommend to draw its sketch like you can see at the left.

As you can see, we drew a simple pendulum and chose two systems of coordinates (SC) - the base frame $OX_0Y_0Z_0$ (SC0) and the body-fixed frame (SC1). The SC0 origin is placed in the center of the joint, the second one (SC1) - at the mass center of the pendulum. The axes of SC1 are directed along the pendulum principle axes of inertia. The base frame exists in every object and, as a rule, is connected with the Earth. There is only rotational joint connecting the pendulum and the base frame (the wall which the pendulum is attached to).

1.3. Creating the model

1.3.1. Running UM Input and creating new model

Running UM Input program

1. Click **Start/Programs/Universal Mechanism 5.0/UM Input**.

Creating a new model

1. From the **File** menu point to **New object**.

The window of the constructor appears, see fig. 2.

1.3.2. Familiarizing yourself with Universal Mechanism

Take a few minutes to familiarizing yourself with the Universal Mechanism constructor window, see Figure 2.

Tree of elements of a model in the left top corner of the constructor window is used for getting access to elements of the model.

Animation window in the center shows the model or its elements. A frame is shown in the center of animation window. There is the following identification for axes: Red – X, Green – Y, Blue – Z (RGB). Point of view, zoom and other settings can be changed via toolbar buttons. Using the context menu you can set perspective parameters, supporting grid, etc.

Inspector at the right-hand side of the constructor is the main tool for the description of elements. It shows parameters of an active element. It contains full information about current element of the model.

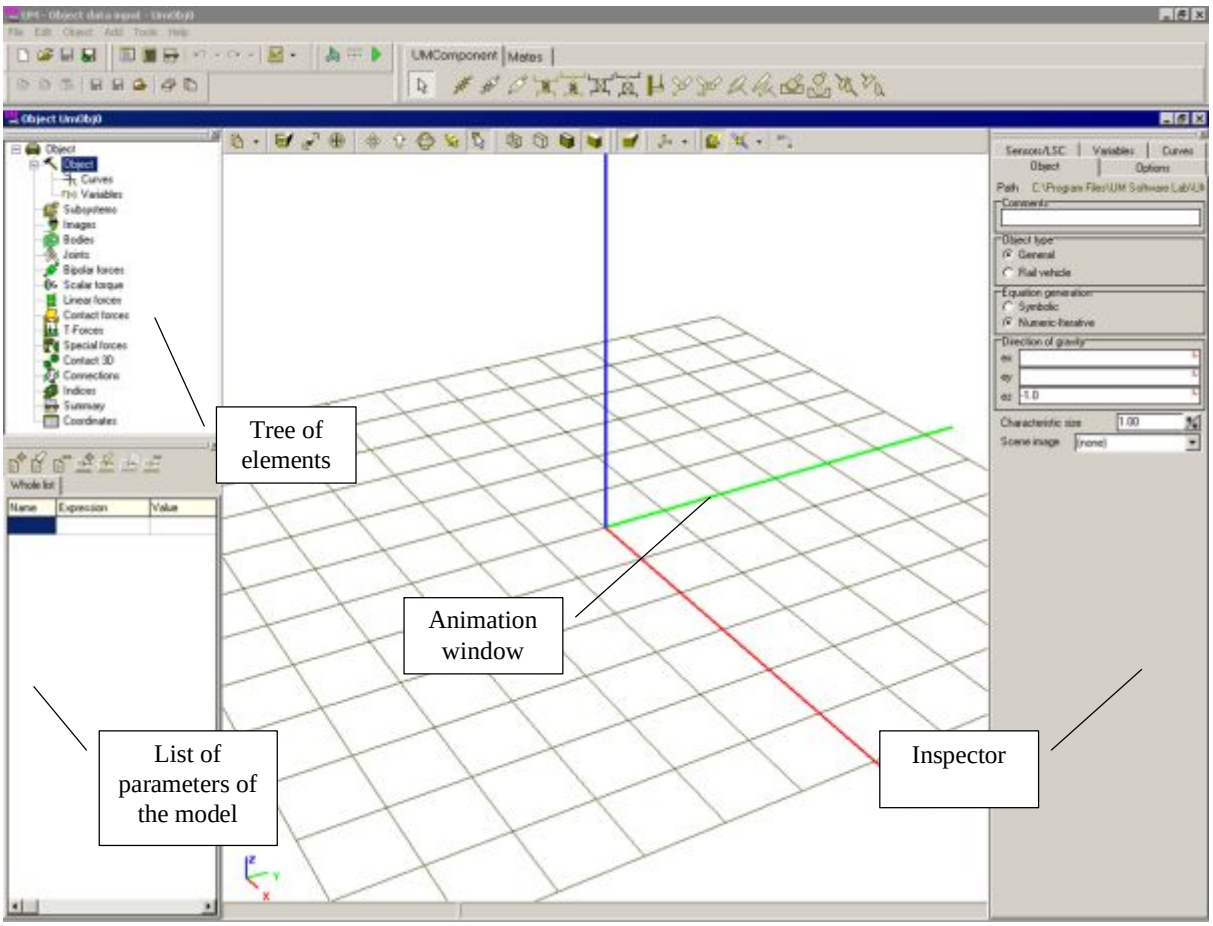


Figure 2. Constructor window

1.3.3. Creating graphical objects

We recommend to start describing any mechanical system with creating a set of graphical objects (GO) of the elements of the model.

1.3.3.1. Scene image

Creating new graphical object - scene

Scene is a graphical object corresponding to fixed elements of the object. Describing the scene is optional. To create a scene you should make a usual graphical object and assign it to the scene image. As for our example it is an image of the fixed joint where the pendulum is attached to – support. In order to create the corresponding image you should do the following steps.

1. Point to **Images** element of **Tree of elements**.
2. Click button in the top of the **Inspector** to create new graphical object, see Figure 3.

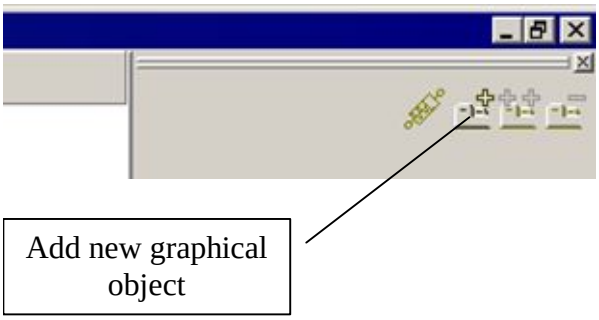


Figure 3. Adding a new element

Note You can add new element of any type in the same way.

Renaming the graphical object

As you create objects, UM automatically assigns names to them. Each name consists of a string containing the element type and a unique integer ID for that type. UM named the recently created graphical element **GO1**.

1. Point to the box with the name of the element and replace **GO1** with **Support**, see Figure 4.

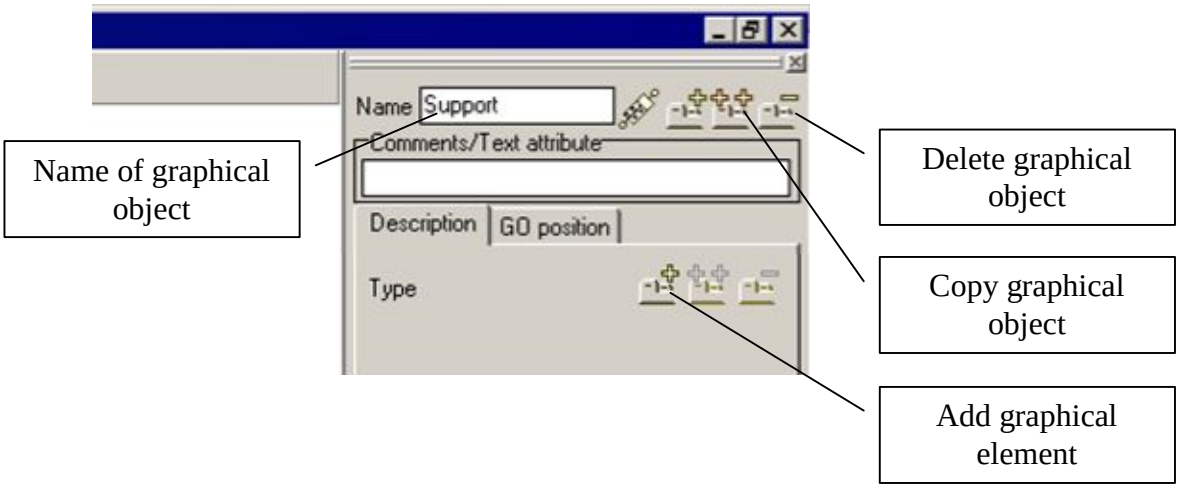


Figure 4. Renaming the graphical object

Creating graphical elements

Every *graphical object (GO)* can include any number of various graphical elements *(GE)*. So you are able to create quite complicated images. Let's create three elements - sphere, cone and box, which form the image of the **support** altogether.

Creating new graphical element: sphere

1. Click **Add new graphical element** button, see Figure 4.

New tab **GE1** appears, see Figure 5.

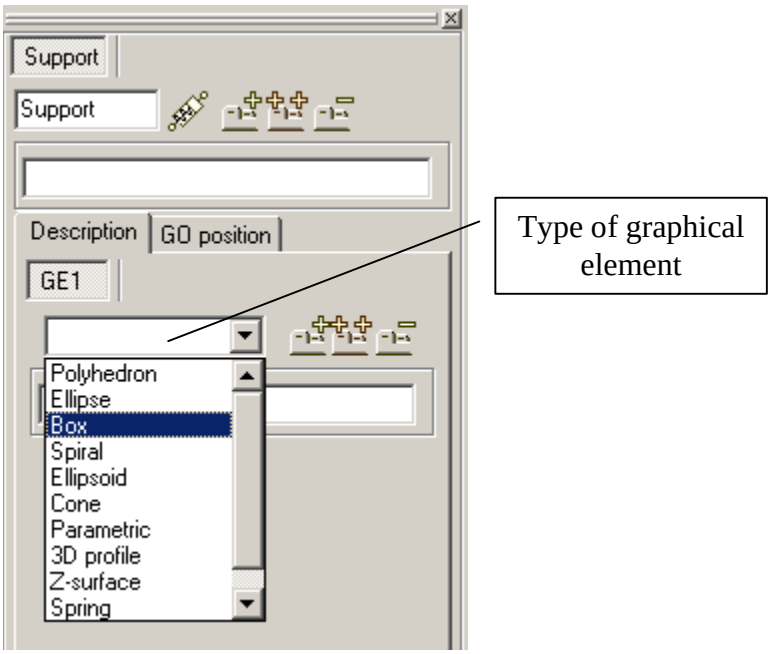


Figure 5. Type of the graphical element

2. Choose type for the new *graphical element* – **Ellipsoid**.
3. Point to the **Parameters** tab and set **a = b = c = 0.05**.
4. Point to the **Color** tab and set diffuse color to red.

Creating new graphical element: cone

1. Create new *graphical element* and set its type to **Cone.**

Note: Do not add new *graphical object* instead new *graphical element* within graphical object. In this example we create the only *graph object* – **Support**, which contains three *graphical elements*: *sphere*, *cone* and *box*.

2. Point to **Parameters** tab and set **R2 = 0.1; R1 = 0; h = 0.15**.
3. Set diffuse color to red.

Creating new graphical element: box

1. Create new graphical element and set its type to **Box**.
2. Point to **Parameters** tab and set **A = 0.5; B = 0.5; C = 0.05**.
3. Point to **GE Position** tab. Set **Translation/Z** to **0.15**, see Figure 6.

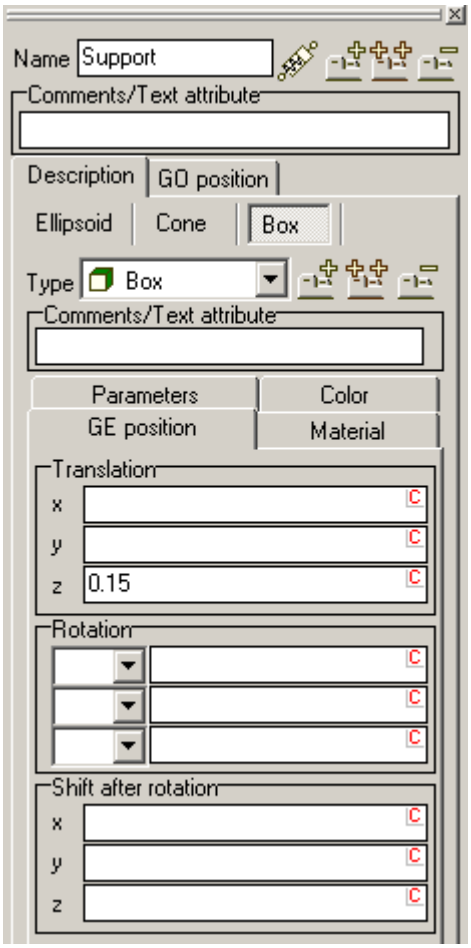
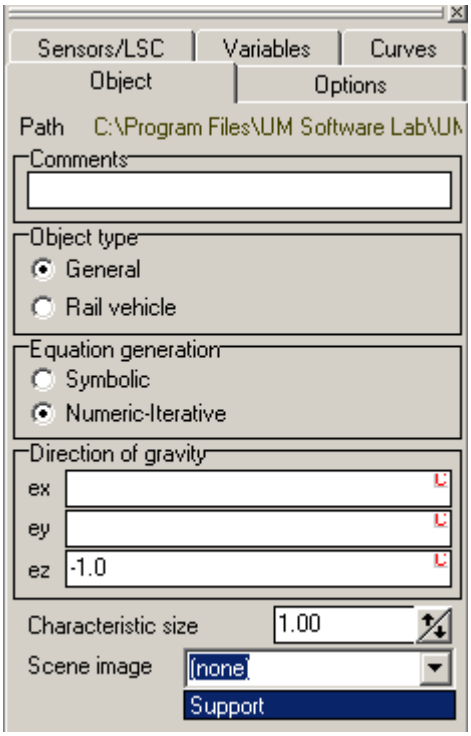


Figure 6. Graphical element position

Assigning Support as scene image

1. Point to **Object** item of **Tree of elements**.
2. Select **Support** in the **Scene image** list.



1.3.3.2. Image of pendulum

1. Return to the **Images** item in the **Inspector**.
2. Create new graphical object.
3. Rename new graphical object to **Pendulum**.

Note: Do not forget to press **Enter** after any modification of the text data in order to reflect this.

The pendulum image consists of two *graphical elements*: an ellipsoid and a cone.

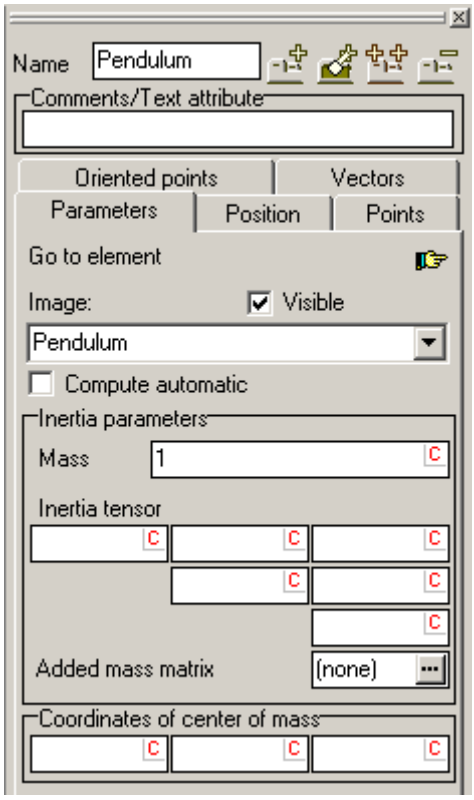
4. Add new graphical element **Ellipsoid** and set its parameters **a = 0.05; b = 0.2**; **c = 0.2.** Set diffuse color to blue**.**
5. Add new graphical element **Cone** and set its parameters **R2 = 0.03; R1 = 0.03; h = 1**. Set diffuse color to blue.

Now image of the pendulum is ready.

1.3.4. Creating rigid bodies

The pendulum as a mechanical system consists of the only body.

1. Point to the **Bodies** item in the **Inspector**.
2. Create new body.
3. Rename body to **Pendulum**.
4. Select **Pendulum** from the drop-down list **Image**.
5. Set **Mass** = **1** (kg).



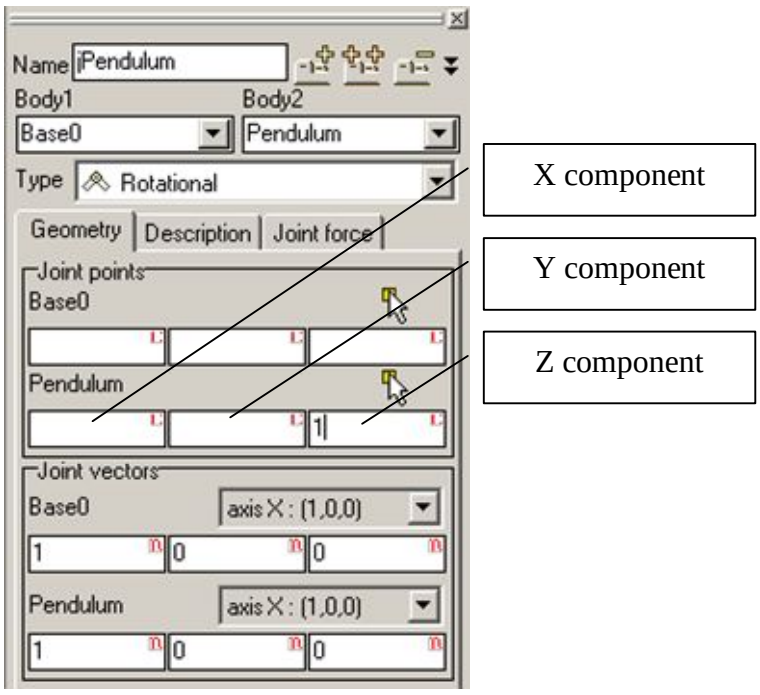
1.3.5. Creating joints

The rotational joint connects the Pendulum and the Base0. To create new joint do the following actions:

1. Point to **Bodies/Pendulum**.
2. Click the button **Adjust joint** and select **Rotational** joint in the context menu.

After that the rotational joint is created and named as **jPendulum** automatically. **Joint points** and **joint vectors** describe the position of the rotation axis relative to each of the bodies. Their coordinates must be given in the corresponding body-fixed systems of coordinates.

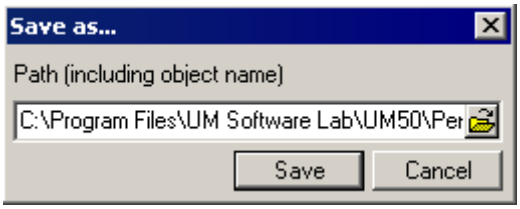
3. In the group **Joint points/Pendulum** set **Z** position to **1**. So the pendulum will swing around its upper point.



1.3.6. Saving the model

Now your model is described completely. And it is high time to save it. Let the object name be **Pendulum**.

1. Select menu item **File/Save as…**
2. Set **Path** to ***{UM Path}*\Pend**, in the way how it is shown in the figure below.



1.3.7. Preparation for simulation: generation of equations of motion

Program package Universal Mechanism (UM) consists of two programs: **UM Input** program – UMInput.exe and **UM Simulation** program – UMSimul.exe.

Now we should prepare our model for subsequent analysis in the **UM Simulation** program. We should choose the method of generation of equations of motion. Universal Mechanism supports two methods: *symbolic* and *numeric-iterative*. Let us consider them more detailed.

Symbolic generation of equations of motion

Symbolic method assumes generation equations of motion as source files in *C* or *Pascal* with posterior their compilation by one of the supported external compilers. As a result of compilation the *UMTask.dll* appears. This *.dll is used by **UM Simulation** program for numerical integration of equations of motion.

Generation and compilation of equations of motions are performed within **UM Input** program.

Numeric-iterative generation of equations of motion

Numeric-iterative method assumes generation of equations of motion on each step of numerical integration directly in **UM Simulation** program.

Comparison of symbolic and numeric-iterative methods

Let us consider advantages and disadvantages of both methods.

In terms of CPU efforts the symbolic method is faster. It provides decreasing CPU efforts up to 10-30% for complex (more than 10-20 degrees of freedom) models. For rather simple models CPU efforts for both methods are roughly the same. The symbolic method during generation of source code fulfils its optimization from the point of view of CPU-efforts.

On the other hand the symbolic method of generation of equations of motion expects any external compiler to be installed on the same computer. Universal Mechanism supports Borland Delphi, Borland C++ Builder, Microsoft Visual C++ as external compilers.

At the same time the numeric-iterative method does not suppose explicit steps of generation and compilation of equations of motion and seems to be simpler in usage.

Recommendations

For beginner users it is strongly recommended to use the numeric-iterative method of generation of equations of motion as simpler in usage. The symbolic method might be recommended for more experienced users which work with more or less complex models.

During the “Getting Started” series we will use the numeric-iterative method as the basic one.

Chapters 1.3.7.1 and 1.3.7.2 contain the descriptions of preparing for simulation sequence in case of usage of the numeric-iterative and symbolic methods of generation of equations of motion consequently.

1.3.7.1. Numeric-iterative method

1. In the tree of elements select **Object**.
2. Set **Generation of equations** to **Numeric-iterative** (see Inspector window in the right part of the constructor window).
3. Save the model again.

Now the model is ready for simulation within the **UM Simulation** program. How to start numerical simulation please read in the Sect. «1.3.8. Run UM Simulation program», page 21.

1.3.7.2. Symbolic method

If you used the numeric-iterative method for generation of equations of motion you may omit this section and come directly to Sect. 1.3.8.

The UM Input is used for creating objects, generating their equations and compiling them with the help of an external compiler. As a result you have got a dynamic-linked library UmTask.dll containing equations of motion of your object. The DLL is always located in the object directory. When DLL of the object exists a model is ready for simulation.

Now we should generate and compile equations of motion and start **UM Simulation** for dynamical analysis of the system.

Paths to external compilers

Universal Mechanism supports using Borland Delphi, Borland C++ Builder, Microsoft Visual C++ as external compilers.

1. Select **Tools/Options** menu item.

Your further actions depend on what external compiler you are going to use:

Delphi

2. Select **Paths/Delphi** tab.
3. Click **Search Delphi** button.

Borland C++ Builder, Microsoft Visual C++

2. Select **Paths/C++** tab.
3. Click one of the following buttons **Search Visual C** or **Search Borland C++ Builder** depending on which C compiler is installed on your PC.

If UM successfully detects external compiler all paths are set automatically.

If the compiler is not found automatically (e.g. if its newest version is not included in the internal UM list of compilers), paths to compiler and libraries should be set manually.

Generating and compiling equations of motion

1. Select **Object/Generate equations**.

If your description of the model is correct the corresponding dialog box appears. If your model description is not correct then tab **Summary**, which contains all detected errors, appears.

2. Select **Compile equations**.
3. Click **Generate** button.

If generating and compiling equations of motions end successfully you'll see the message box: «**Compiling successful. Object is ready for simulation.**». The model is ready to be loaded in the **UM Simulation** program.

1.3.8. Run UM Simulation program

1. Select **Object/Simulation** menu item.

UM Simulation program starts and opens the current model.

1.4. Simulation of motion

Now we are in the simulation program. We will open new animation window, deflect the pendulum from vertical position to **1** radian and run simulation of dynamics of pendulum.

Creating new animation window

1. From the **Tools** menu, select **Animation window**. New animation window appears. Familiarize yourself a bit with an animation window.

Rotating

Point the mouse cursor to the animation window so that cursor looks like the picture in the figure to the right. Press left mouse button and rotate the model in the animation window.

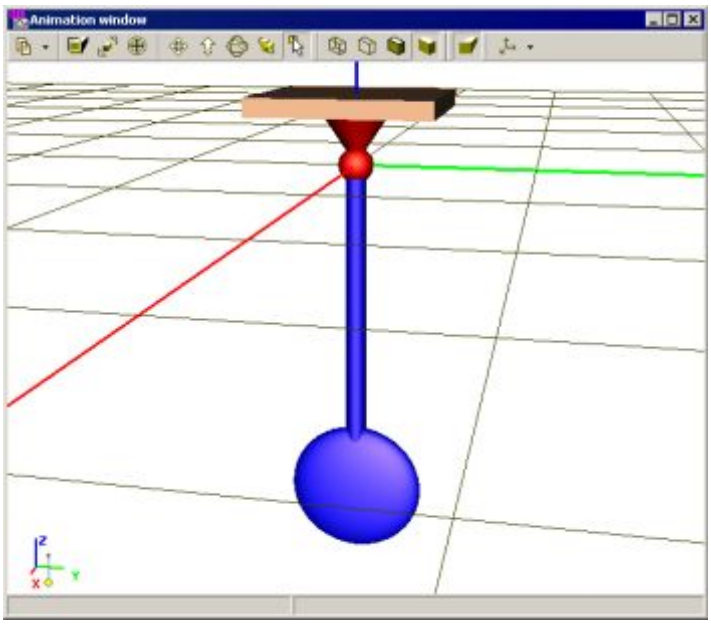
Shifting

Point the mouse cursor to the animation window so that it has *Rotating* shape, press **Ctrl** key and mouse cursor changes to *Shifting* mode. Press left mouse button and shift model in the animation window.

Zoom in/zoom out

Point the mouse cursor to the animation window and press **Shift** key and with the help of left mouse button zoom in/out the model. You can also use mouse scroll wheel.

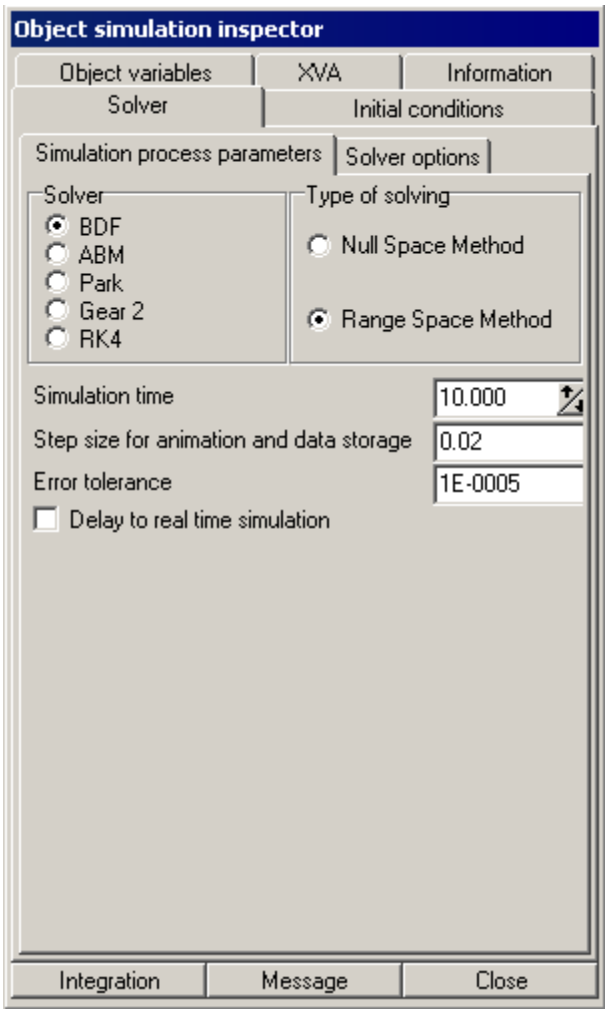
After some practice you can get something like shown in the figure below.



Start simulation

1. From the **Analysis** menu, select **Simulation**.

Window of the **Object simulation inspector** appears.



Initial conditions

You should deflect the pendulum a bit in order to obtain its motion. There exists a special tool for this purpose: a wizard of the initial conditions.

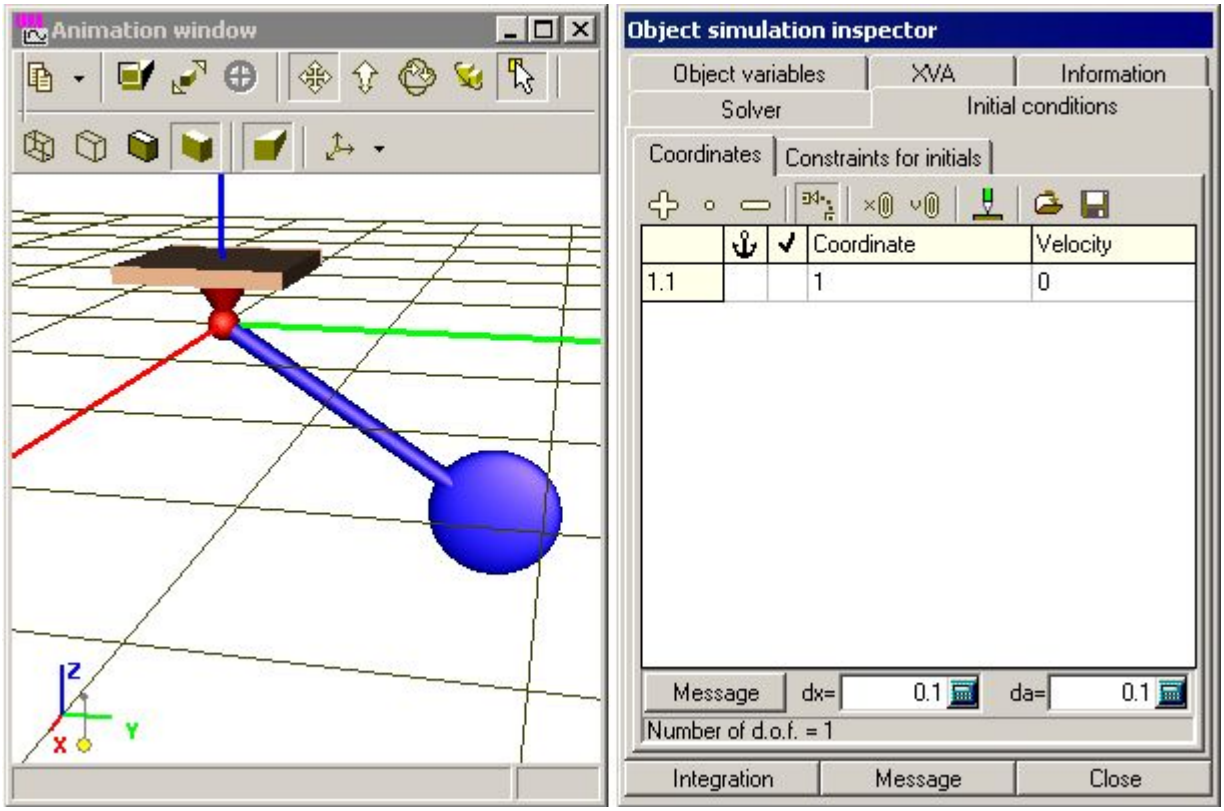
1. Select the **Initial conditions | Coordinates** tab.

You can see a complete list of the object coordinates. In our case there is only one coordinate in **jPendulum** joint.

2. Set **Coordinate** to **1**. Press **Enter** key.

Your pendulum has deflected.

Note: Universal Mechanism uses *System International (SI).* Angular values have dimensions of *radian*.

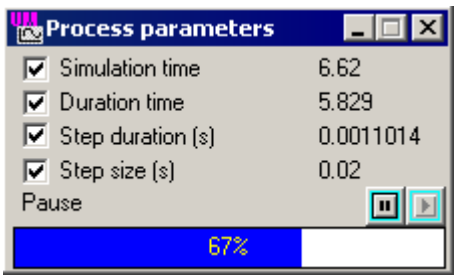


Simulation

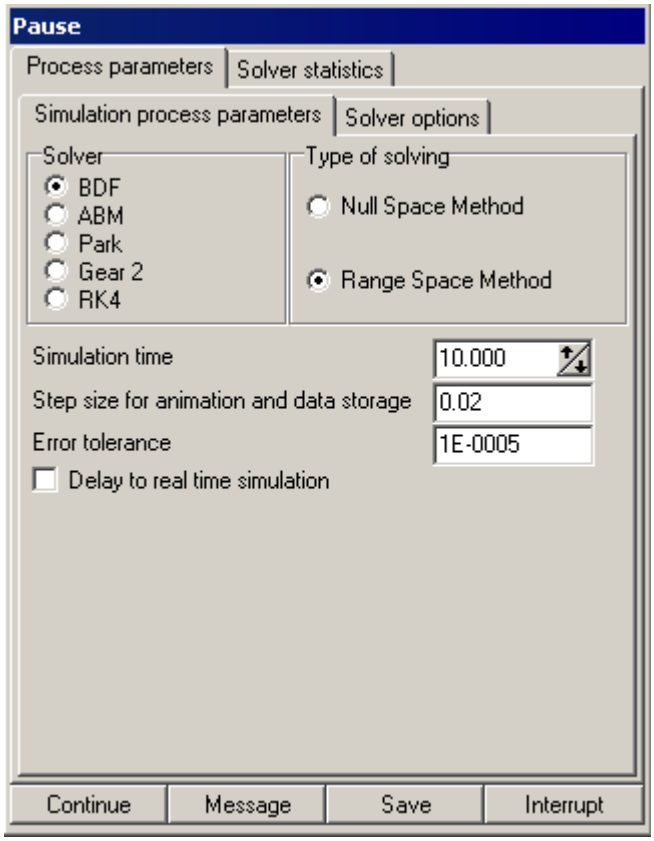
Now your model is ready for simulation. Simply start simulation process for the **10** seconds.

1. Click **Integration** button in the **Object simulation inspector**.

Process parameters window will appear at the lower right corner of the screen.



At the end of the simulation the **Pause** window appears. You can increase the simulation time, change the numerical method etc.



2. Press the **Interrupt** button. **Object simulation inspector** appears.

Drawing plots

During the simulation you can see plots of various variables. Such as velocities, accelerations, forces and so on. We will open new graphical window, create new variable to plot – Y coordinate of the center of mass of the pendulum and draw its plot.

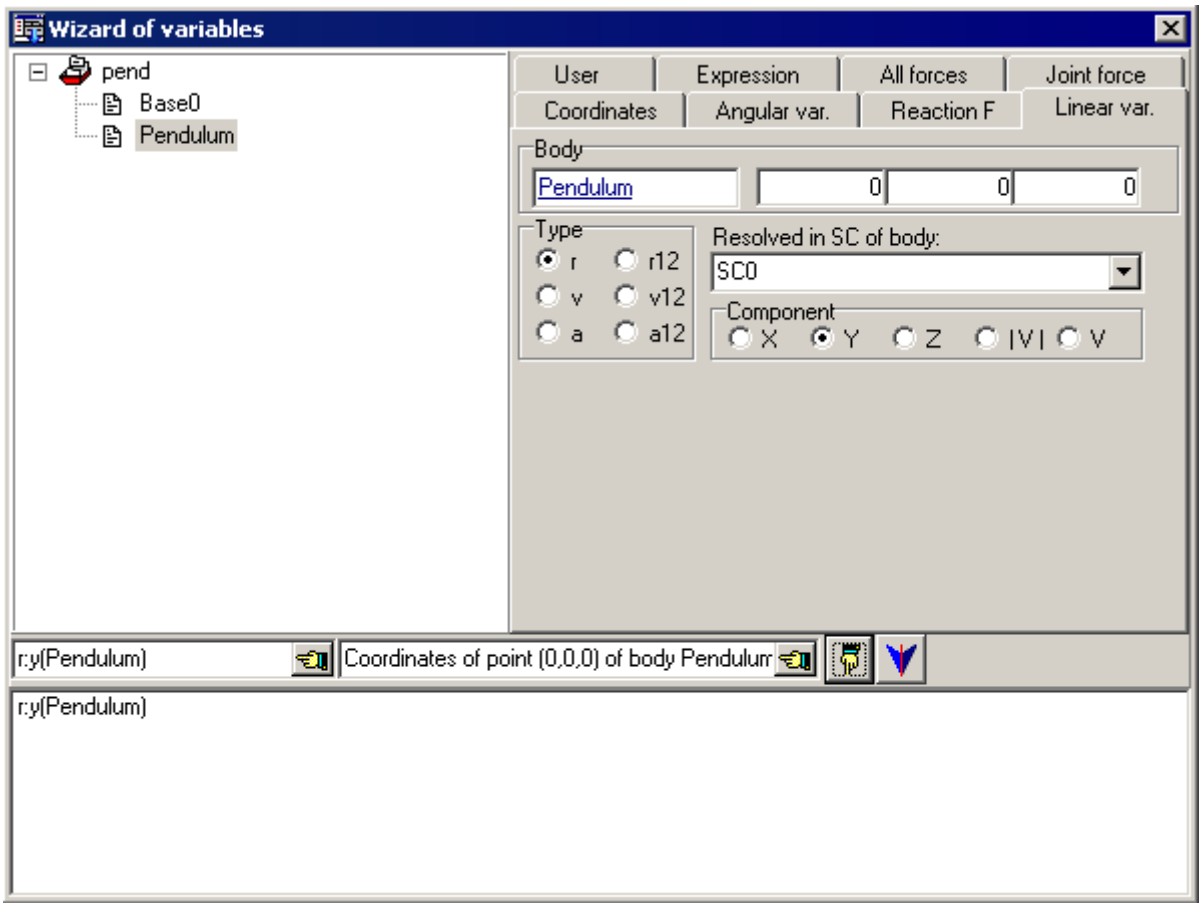
Well, let us create new graphical window.

1. From the **Tools** menu, select **Graphical window**.

Open **Wizard of variables**.

2. From the **Tools** menu, select **Wizard of variables**.

The **Wizard of variables** is a special tool for creating variables, which can be drawn in graphical windows or animated in animation windows (in cases of vectors or trajectories).



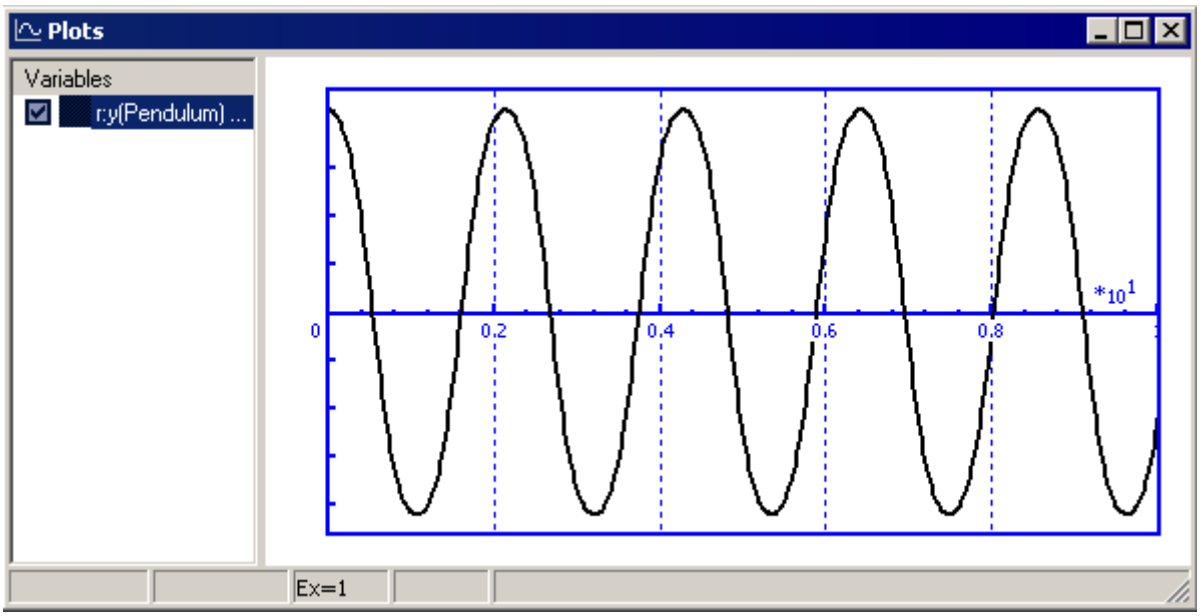
Let us draw a plot of Y coordinate of the mass center of the pendulum.

3. Select **Linear var.** tab (linear variables: coordinates, velocities, accelerations etc.).
4. Select **Y** in the **Component** group.
5. Then move the variable to the container with the help of the button .

New variable **r:y(Pendulum)** appears in the container of variables.

6. Select the variable in the **Wizard of variables** and drag it to the graphical window.
7. Select the **Object simulation inspector** and click the **Integration** button.

You can see the plot of your variable in the graphical window.



Animation of vectors and trajectories

During the simulation you can animate various vector variables in an animation window. Let us animate the vector of the mass center velocity. Firstly, we need to create such variable in the **Wizard of variables**.

1. Select the **Wizard of variables** and there select the **Linear var.** tab.
2. Select **v** (velocity) in the **Type** group.
3. Select **V** (vector) in the **Component** group.
4. Add this variable to the container clicking the button.
5. Drag new variable to the animation window.

A list of animated vectors is hidden by default. You can make it visible and change its position with the help of the **Position of list of vectors** command of the pop up menu of the animation window.

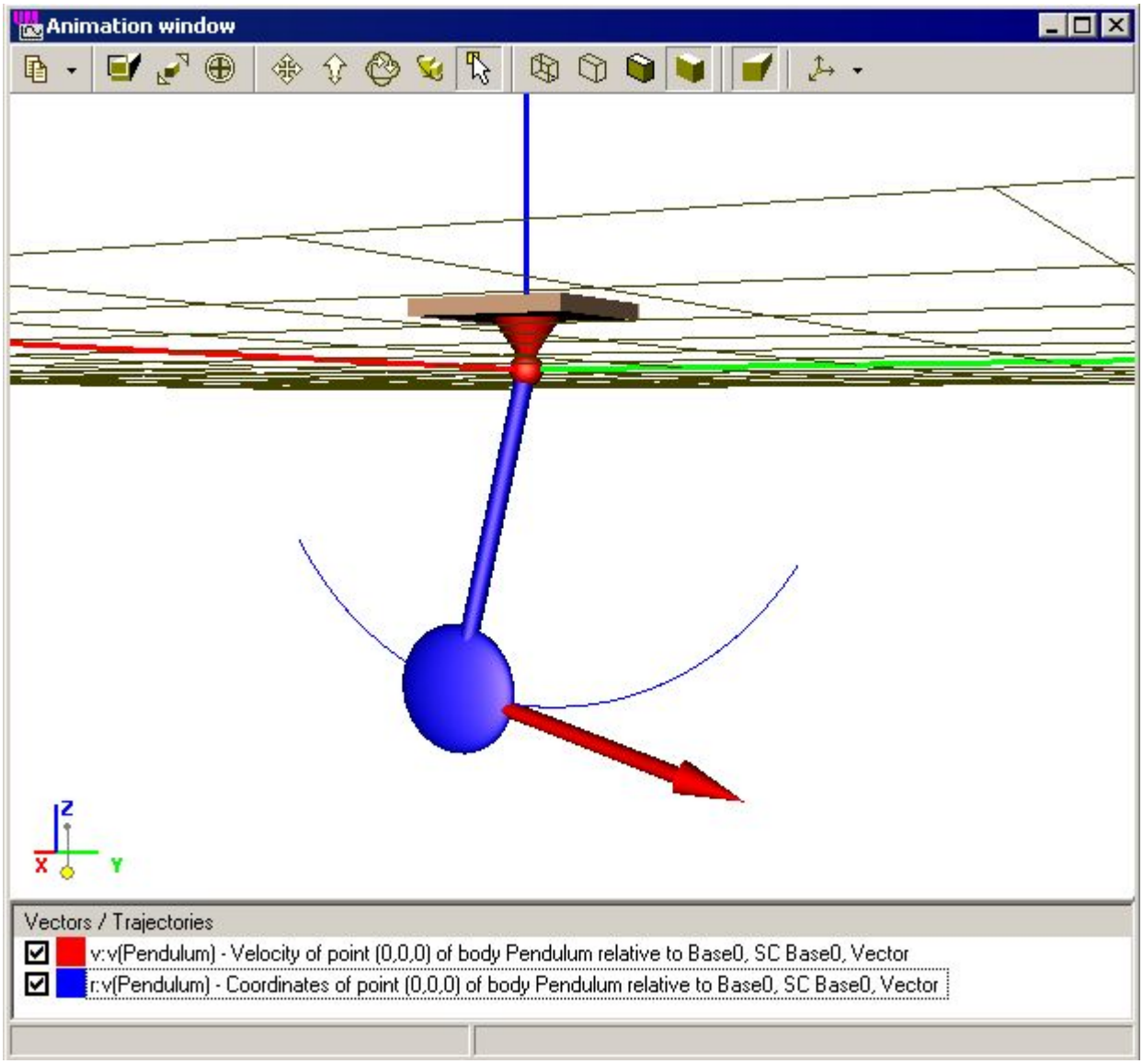
6. Select animation window, click right mouse button and select **Position of the list of vectors/Left**.

To draw a trajectory of the pendulum create a new variable with the help of the master.

7. Repeat all steps we made for the velocity, but the **Type** of the variable set to **r** (radius-vector). Drag this variable to the animation window.
8. Double click on the velocity item in the **List of vectors** and select red color for the vector of velocity and than double click trajectory item and select blue color for it.
9. Click the **Integration** button in the **Object simulation inspector**.

Now you can see the vector of the velocity and trajectory of the center of the mass of the pendulum. You should use the **Scale of vectors** command of a pop up menu to specify its scale.

Double click on an element of the list of vectors or on a vector/trajectory image to change the color of the vector and trajectory (in addition for the trajectory - to change the number of points on the curve).

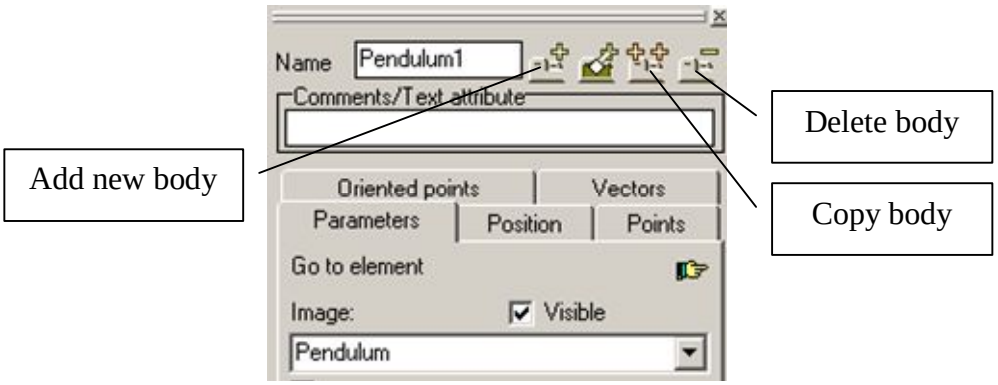

Animation window
Z
X
Y
Vectors / Trajectories
v:v(Pendulum) - Velocity of point (0,0,0) of body Pendulum relative to Base0, SC Base0, Vector
r:v(Pendulum) - Coordinates of point (0,0,0) of body Pendulum relative to Base0, SC Base0, Vector

1.5. Multibody pendulum

We can use two approaches to develop multi-link pendulum (pendulum chain) from the simple pendulum model. First variant is the development of the simple pendulum model with copying of the existed elements. Second variant is based on the usage of subsystem technique.

1.5.1. Development of the simple pendulum model

1. Close the **UM Simulation** program and come back to the **UM Input** program.
2. Select **Bodies** and copy the pendulum two times.

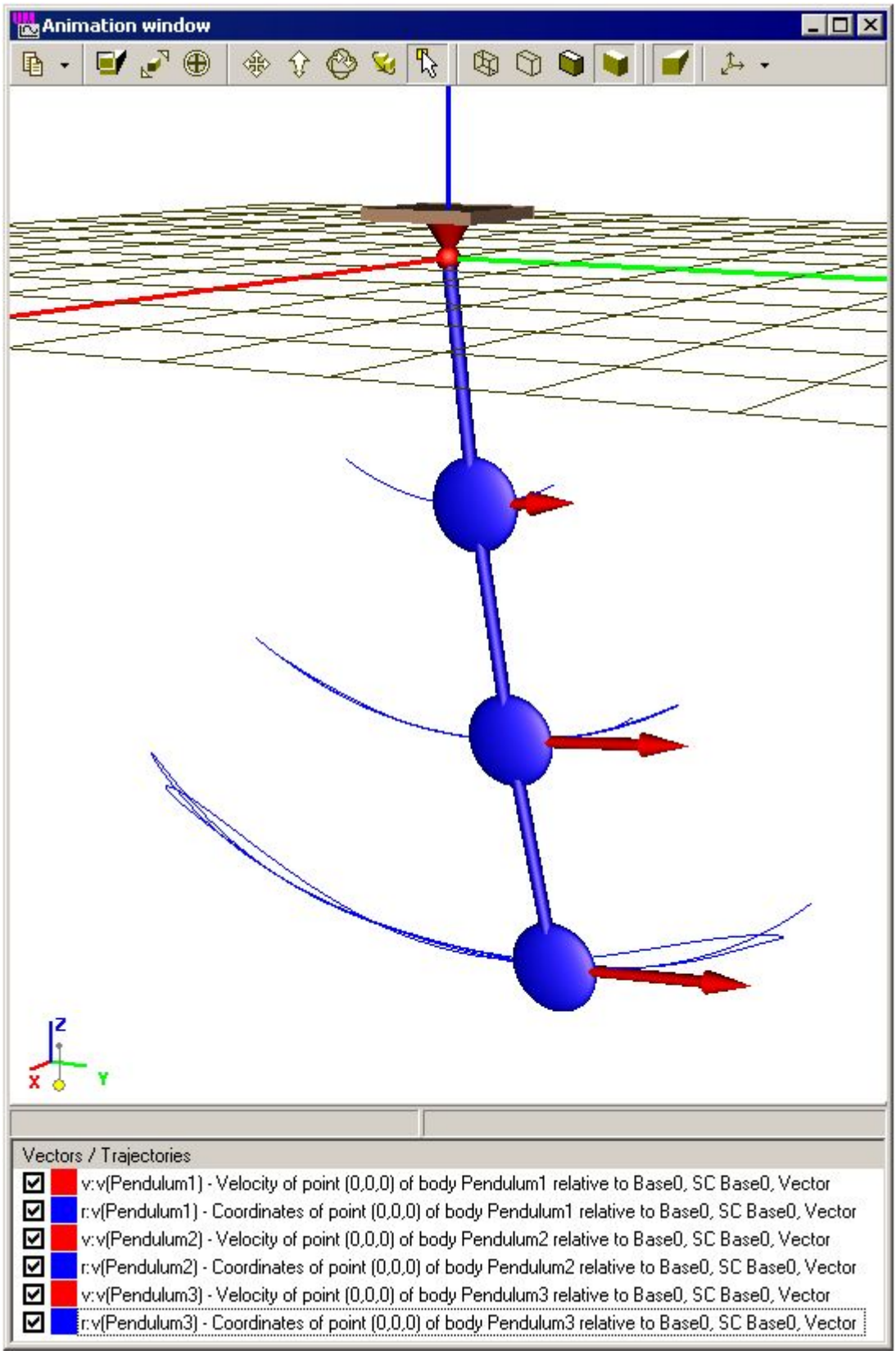


3. Rename body **Pendulum** to **Pendulum1** and new bodies to **Pendulum2** and **Pendulum3**.
4. Select **Joints** and copy **jPendulum** joint two times too.
5. Change the connecting bodies: **Pendulum1** and **Pendulum2** for the second joint, **Pendulum2** and **Pendulum3** for the third one.

Note: Use the button in the top of the animation window to switch the mode of window Full object / Single element.

6. Save the model as **ThreeLinkPend** (main menu item **File | Save as…**).
7. Generate equations of motion with numerical-iterative method (Sect. 1.3.7.1, page 19).
8. Run UM Simulation.
9. Create a new animation window.
10. Select the **Analysis/Simulation** menu item.

11. Set a proper initial position of the chain with the help of the **Initial conditions** tab. Set **0.3** for all the model coordinates.
12. Click **Integration** to run the simulation.


Animation window
Z
X
Y
Vectors / Trajectories
v:v(Pendulum1) - Velocity of point (0,0,0) of body Pendulum1 relative to Base0, SC Base0, Vector
r:v(Pendulum1) - Coordinates of point (0,0,0) of body Pendulum1 relative to Base0, SC Base0, Vector
v:v(Pendulum2) - Velocity of point (0,0,0) of body Pendulum2 relative to Base0, SC Base0, Vector
r:v(Pendulum2) - Coordinates of point (0,0,0) of body Pendulum2 relative to Base0, SC Base0, Vector
v:v(Pendulum3) - Velocity of point (0,0,0) of body Pendulum3 relative to Base0, SC Base0, Vector
r:v(Pendulum3) - Coordinates of point (0,0,0) of body Pendulum3 relative to Base0, SC Base0, Vector

1.5.2. Usage of subsystem technique

Subsystem technique is used for creation of *compound* models of mechanical systems with the help of standard and user-created subsystems. Each subsystem is a mathematical model of the part of the mechanism.

We will add the model of three-link pendulum (**TLP** – **T**hree**L**ink**P**endulum) three times as subsystems to a new object to create a nine-link pendulum model.

Creation of compound objects from included subsystems is considered in this section.

1.5.2.1. Compound object structure

To create the compound model of multi-link pendulum we should define its structure and way of description of interaction between subsystems.

External and *included* subsystems

There are two variants of user-created subsystems supported by «Universal Mechanism» software. *External* subsystem creates the link to the existed model. *Included* subsystems, unlike the external ones, belong to the compound object. The object owns their structures (bodies, joints, forces, etc.) and parameters.

The most effective way to create nine pendulum model is the usage of the three-link pendulum model as an external subsystem, as soon as all subsystems are kinematically identical. On the other hand, usage of included subsystems for compound models creation is more common. Hereupon we will use *included* subsystems in our example.

Description of interaction between subsystems

Two methods can be used to connect bodies of different subsystems with joints or force elements:

1) create necessary element (joint or force element) in the main (compound) object;

2) use special body, called **External**, for the description of elements in subsystems. In the second method, it is necessary to point bodies and characteristic points, called *connection points*, of these bodies for joints and force elements which use external bodies. Setting a connection means to point which body is external for considered element and which point of this body is characteristic for the element.

At the first stage of creation of nine-link pendulum model we will connect subsystems with joints belong to the main (compound) object (Fig.1.7 A). Then we will make it by the means of body **External** and connection points (Fig.1.7 B).

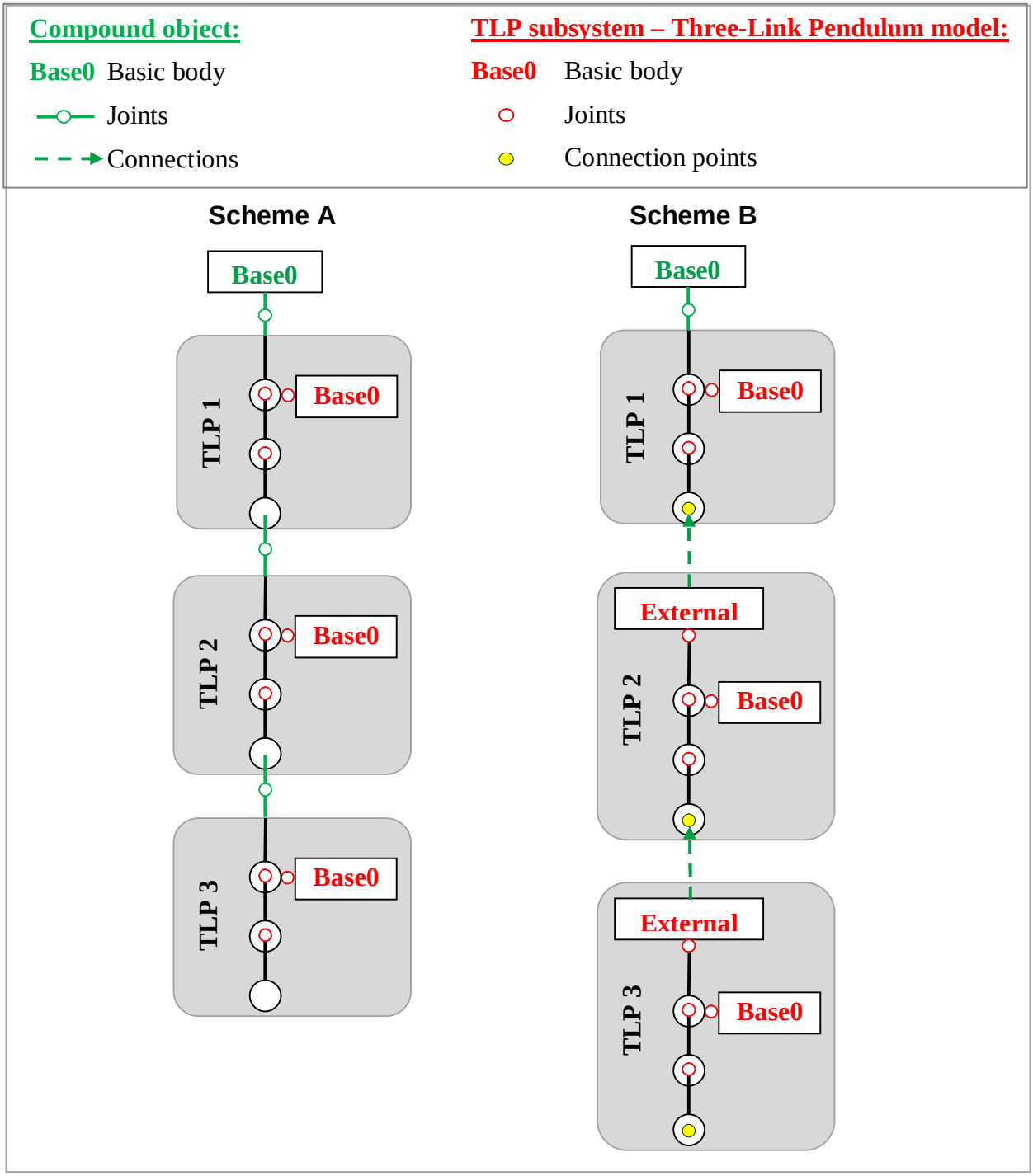


Fig. 1.7. Structure of the compound model of nine-link pendulum

1.5.2.2. Subsystem preparation

We should modify the existed model of three-link pendulum (**ThreeLinkPend** object) to use it as subsystem of compound model.

Modification of kinematics

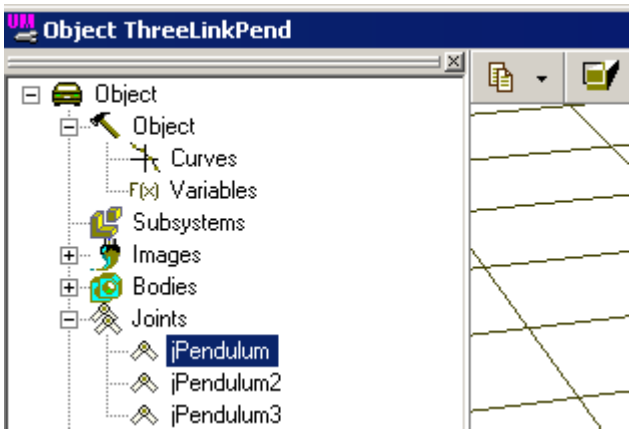
As soon as we will describe rotational joint of the nine-link pendulum suspension as an element of the compound model, and rotational joints linking subsystems will be an elements of compound model (fig. 1.7, scheme A) or will be described in subsystems as joints with **External** body (fig. 1.7, scheme B), joint **jPendulum** is not necessary in subsystem model.

Connectivity condition is necessary for any model of mechanical system. It means that there must be a chain for each body that connects it with body 0 (Base0, SC0). Subsystem kinematics should be matched with kinematics of compound object. As soon as bodies of multi-link pendulum are in planar motion, three-link pendulum model used as subsystem should enable motion of bodies in oscillation plane.

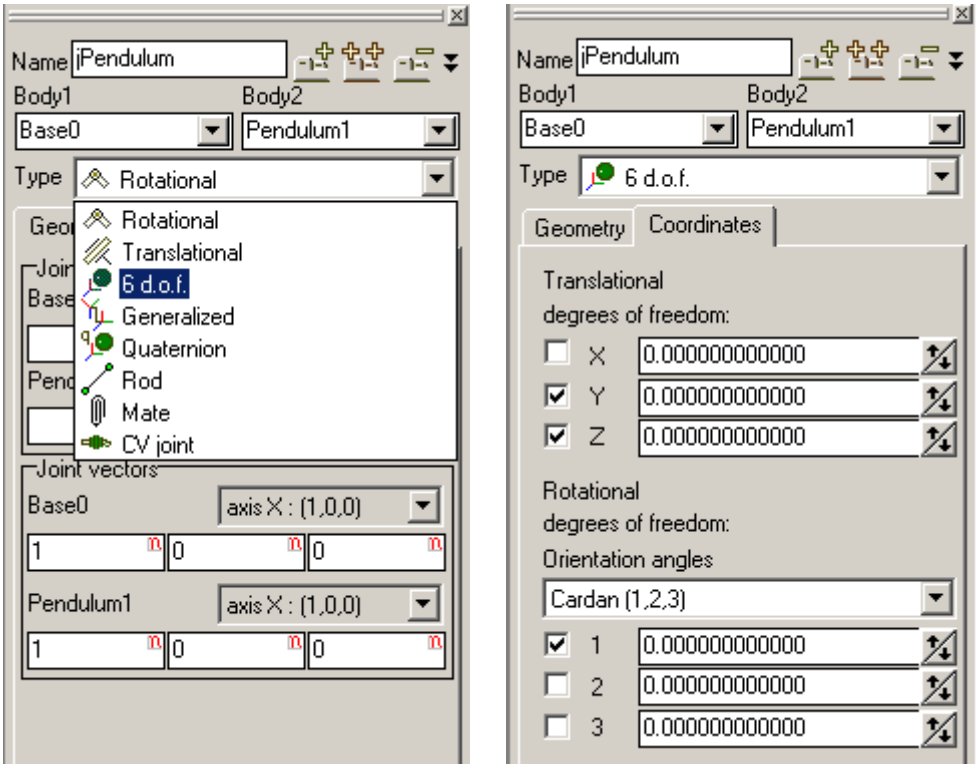
To enable mentioned conditions we will replace rotational joint **jPendulum** in **ThreeLinkPend** model with 3 degree of freedom joint, that permits planar motion of mechanical system in oscillation plane.

1. Close the **UM Simulation** program and come back to the **UM Input** program. The **ThreeLinkPend** model is available.
2. Come to **Joints** element in the element tree and select **jPendulum** joint.

jPendulum joint describes rotational degree of freedom between basic body **Base0** and **Pendulum1** body.



3. Change **Type** of the joint to **6 degree of freedom joint**.



4. Switch off 3 degrees of freedoms as shown at the left figure.

Now, **Pendulum1**, **Pendulum2** and **Pendulum3** can fulfill planar motion in oscillation plane.

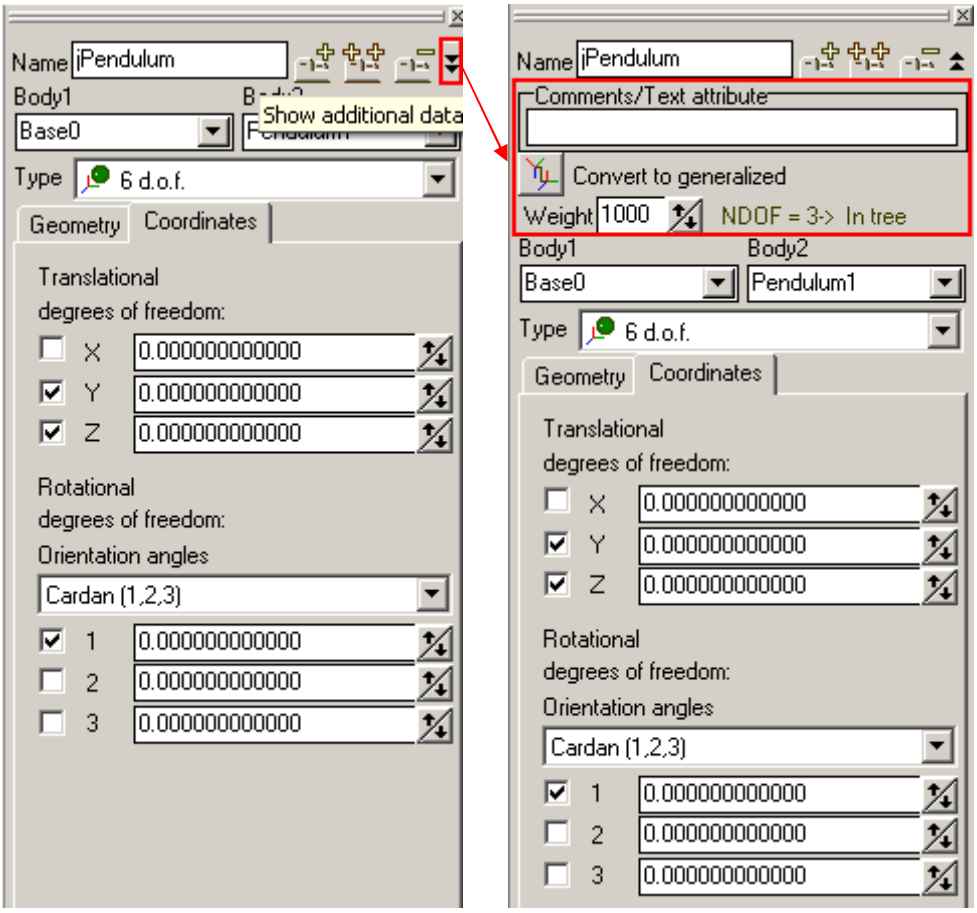
Joints cutting settings

Adding of the subsystems and description of joints between them will result in creation of several *closed kinematics loops* in model kinematics graph. In other words the number of model joint coordinates will be grater then the number of degrees of freedom of the multi-body system.

To remove closed kinematics loops we should cut a few joints. The number of joints to cut equals that of independent cycles in the graph. As soon as the automatic choice of such joints is ambiguous and can result in non-optimal model performance, the user has an ability to select by himself.

Coordinates of the rotational joints between the bodies are the most suitable for the description of the multi-level pendulum model. Therefore, we should select **Weight** value for the joints connecting the first body of each of the subsystems with the **Base0** in a special way to “explain” program that we want to cut them.

5. Click ▼ to show/hide a panel of addition joint data.



6. Set **Weight** to **1000**. If a large weight (e.g. **1000**) is assigned to a joint in a closed loop, it will be cut.
7. Save model as **ThreeLinkPend_Subs** (**File | Save as**).

The improvement of the three-body pendulum model is completed. Now it can be used as a subsystem of nine-body pendulum compound model according to A scheme (Fig. 1.7 A, page 33).

1.5.2.1. Compound model creation

Creating a new model

1. Close the **UM Simulation** program and come back to the **UM Input** program.
2. Don't close **ThreeLinkPend_Subs** object! Create new object (**New object** of the **File** menu) and save as **NineLinkPend**.

Then we will switch over with these two objects (**ThreeLinkPend_Subs** and **NineLinkPend**) necessarily.

1.5.2.1.1. Scene image creation

Let's prepare a scene image for the nine-level pendulum model. We will copy **Support** image from **ThreeLinkPend_Subs** model to **NineLinkPend** model.

As soon as the both objects are now open in **UM Input** program we will use **Copy to clipboard / Paste from clipboard** tools. One can also use **Copy to file / Paste from file** tools to fulfill the coping action. These tools are available for coping of any element of a model in **UM Input** program.

Choice of active object / Switching between program windows

The **List of windows** tool (**Tools | List of windows** or *Alt+0*) is used for switching between program windows: object description windows in **UM Input** program and in **UM Simulation** program.

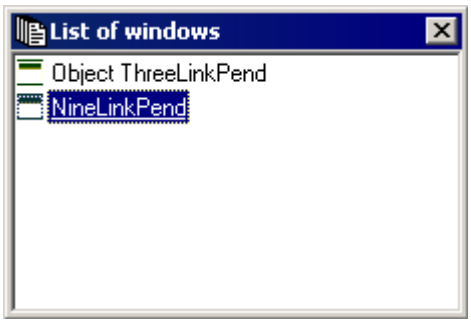


Fig.1.8. Switching between windows of object description in UM Input program

Copy/Paste of object elements

1. Activate **ThreeLinkPend_Subs** object (see Fig.1.8, page 37).
2. Come to **Images** element of tree of elements and select **Support** element.
3. Copy the **Support** to clipboard (**Edit/Copy to clipboard** main menu command).

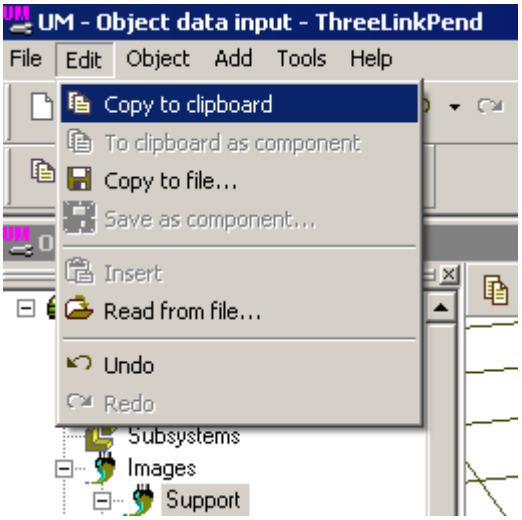
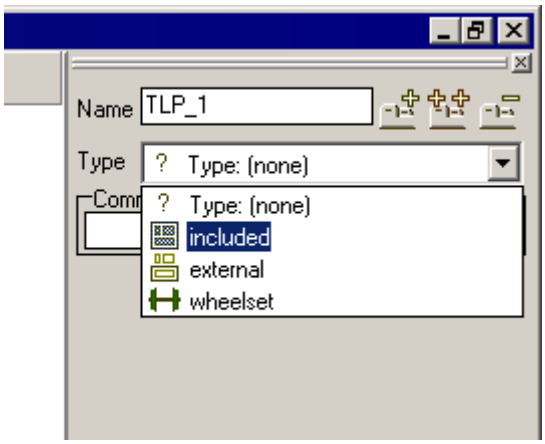


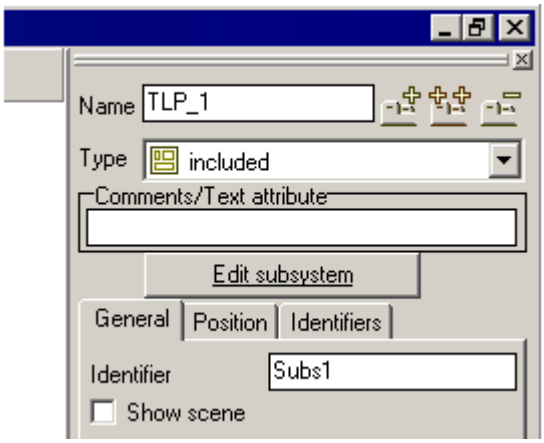
Fig.1.9. Copy to clipboard

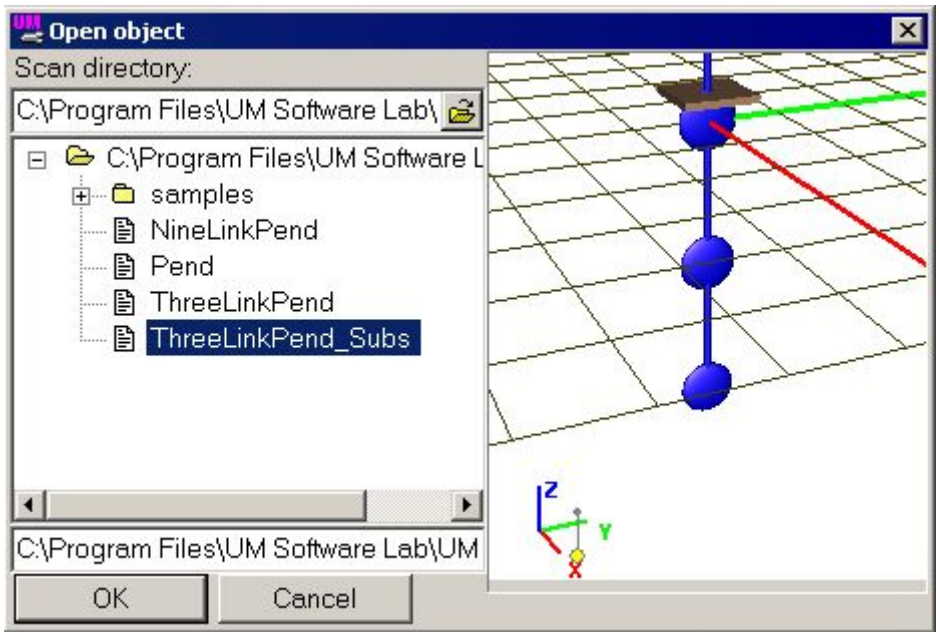
4. Activate **NineLinkPend** object (see Fig.1.8, page 37).
5. Paste **Support** image element from clipboard (**Edit | Paste**). New element will appear in list of **Images**.
6. Select **Object** in element list and select **Support** in the **Scene Image** drop-down list.

1.5.2.1.2. Subsystems adding

1. Activate **NineLinkPend** object (see Fig.1.8, page 37).
2. Come to branch **Subsystems** in the model element tree, add the new element and rename it to **TLP_1 (T**hree**L**ink**P**endulum**_1)**.
3. Select **Subsystem type – Included**. Object open dialog window will appear. Press **F5** key to refresh the list of files in current directory. Select **ThreeLink-Pend_Subs** model and click **OK**.



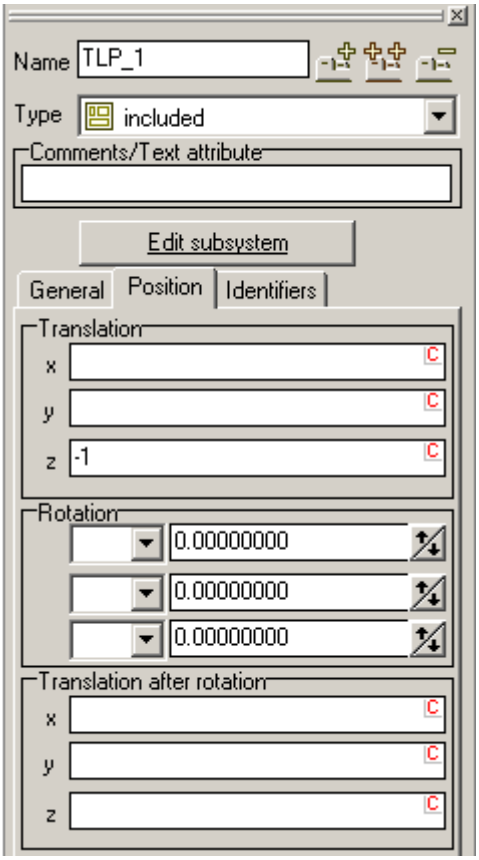


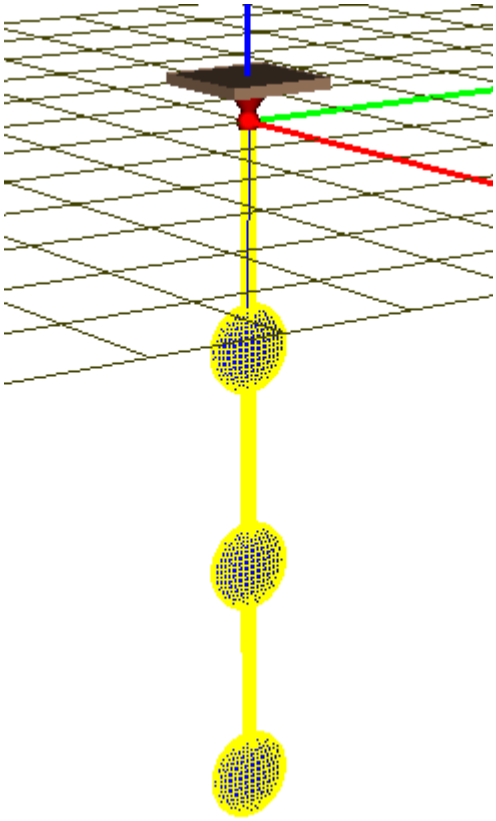


Setting of subsystem position

Subsystem position is evaluated from kinematic constraints defined for its bodies. Nevertheless, the user can preliminary set the position of subsystem in **UM Input** program for better visualization.

4. Point to **Position** tab and set **Z Translation** to **-1** as in the figure below.





5. Copy **TLP_1** subsystem two times.
6. Rename new subsystems to **TLP_2**, **TLP_3**.
7. Set **Z Translation** of **TLP_2** to **-4** and **Z Translation** of **TLP_3** to -7.

Subsystems are added. Now we will create joints to describe interaction between the subsystems into nine-body pendulum model.

1.5.2.1.3. Creating joints

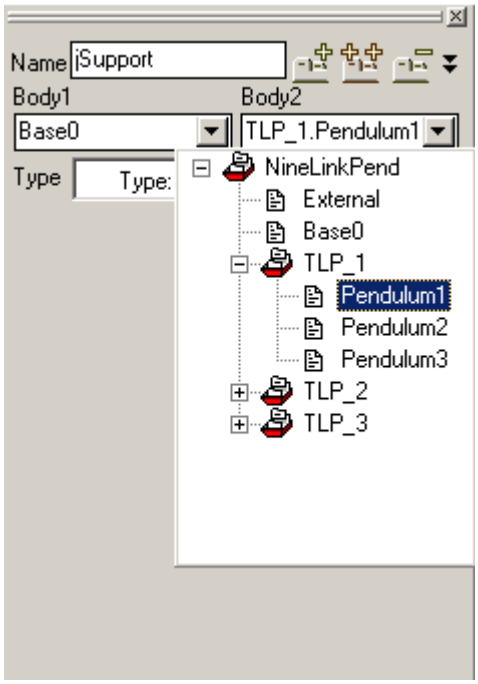
NineLinkPend now includes three subsystems. Each subsystem contains three simple pendulums connected in series with rotational joints with 1 d.o.f. To create nine-body pendulum kinematics we should set rotational joints of the pendulum support and rotational joints between the subsystems.

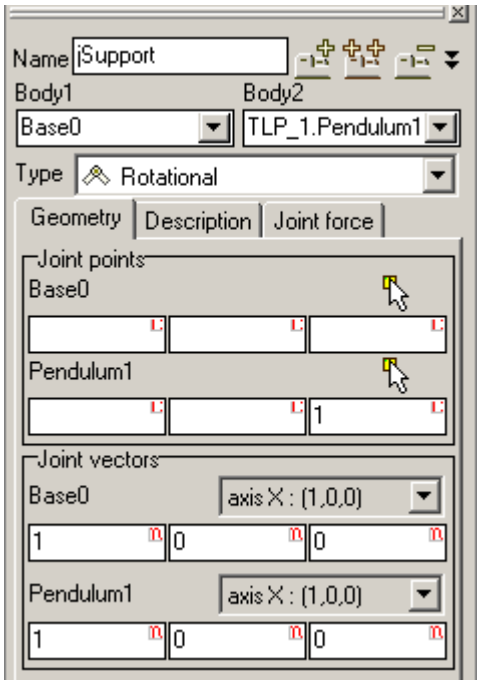
1. Come to **Joints** element in the tree of elements.

Creating pendulum support joint

The rotational joint connects the **Pendulum** and the **Base0**. To create new joint do the following actions:

1. Create a new joint. Let the joint name be **jSupport**.
2. Select the **Base0** as **Body1** and the **TLP_1.Pendulum1** as **Body2**.





3. Set **Type** for the joint – **Rotational**.
4. In the group **Joint points/Pendulum1** set **Z** position to **1**. So the pendulum will swing around its upper point.

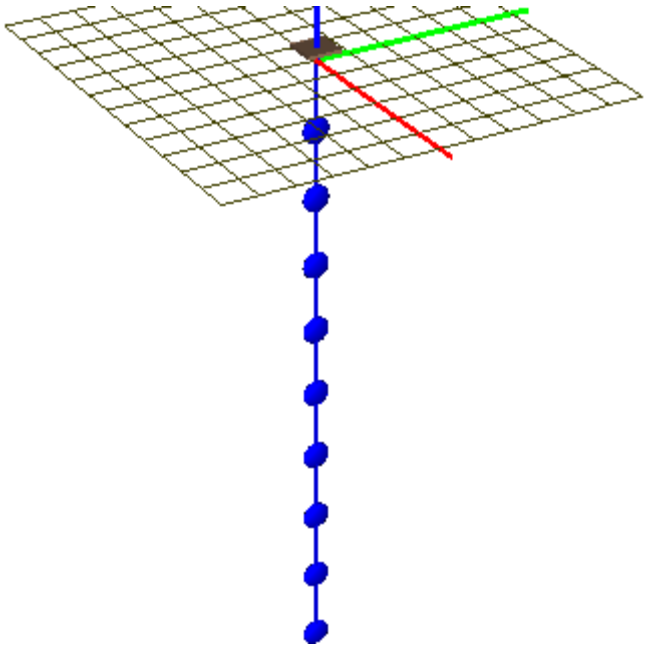
Creating joint between the TLP_1 and TLP_2 subsystems

5. Copy the **jSupport** joint and rename it as **jTLP_12**.
6. Select the **TLP_1.Pendulum3** as **Body1** and the **TLP_2.Pendulum1** as **Body2** for the **jTLP_12** joint.
7. Set **Type** for the joint – **Rotational**.
8. In the group **Joint points/Pendulum1** set **Z** position to **1**.

Creating joint between the TLP_2 and TLP_3 subsystems

9. Copy the **jTLP_12** joint as **jTLP_23**.
10. Select the **TLP_2.Pendulum3** as **Body1** and the **TLP_3.Pendulum1** as **Body2** for the **jTLP_12** joint.
11. Set **Type** for the joint – **Rotational**.
12. In the group **Joint points/Pendulum1** set **Z** position to **1**.

13. Save the model (**NineLinkPend**).



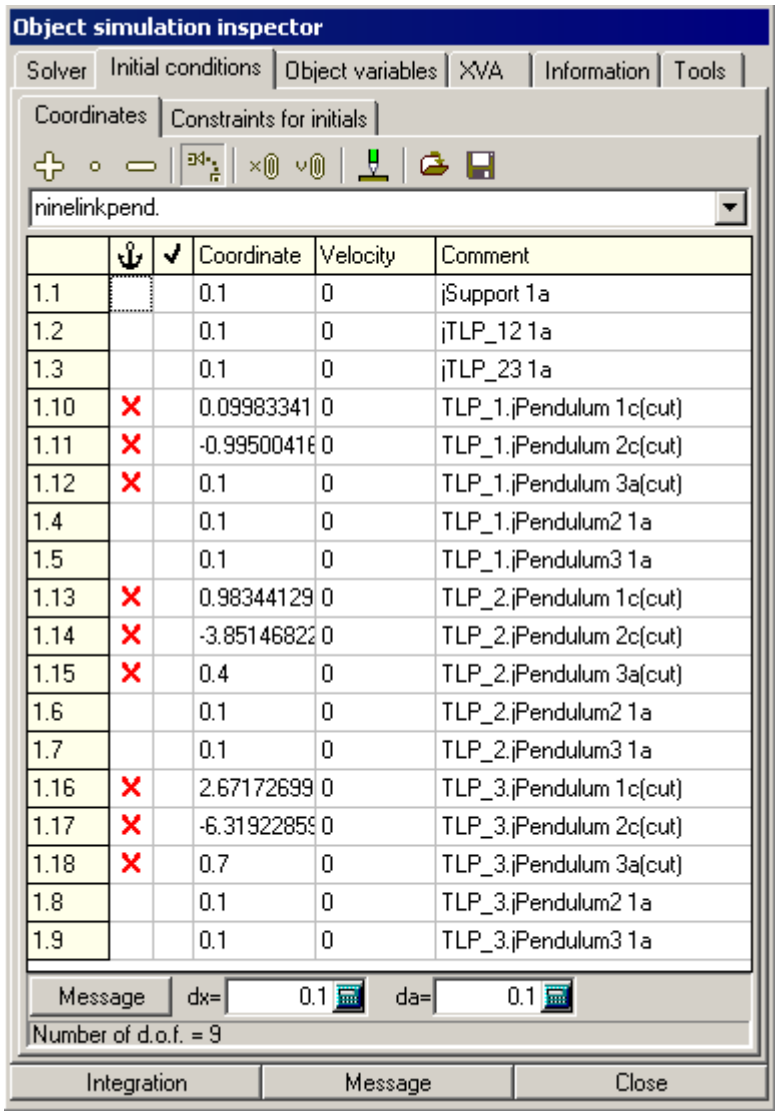
By default the numerical-iterative method of generation of equations of motion is used, (Sect. 1.3.7.1, page 19). Compound model description is now finished. Now it corresponds to scheme A (see Fig. 1.7, page 33). Let's start simulation and save some results.

1.5.2.2. Simulation

Preparation for simulation

1. Run **UM Simulation** (**Object | Simulation…**).
2. Open the **Object simulation inspector** (**Analysis | Simulation…**).
3. Point to the **Initial conditions | Coordinates** tab and set 0.1 value for all model coordinates.

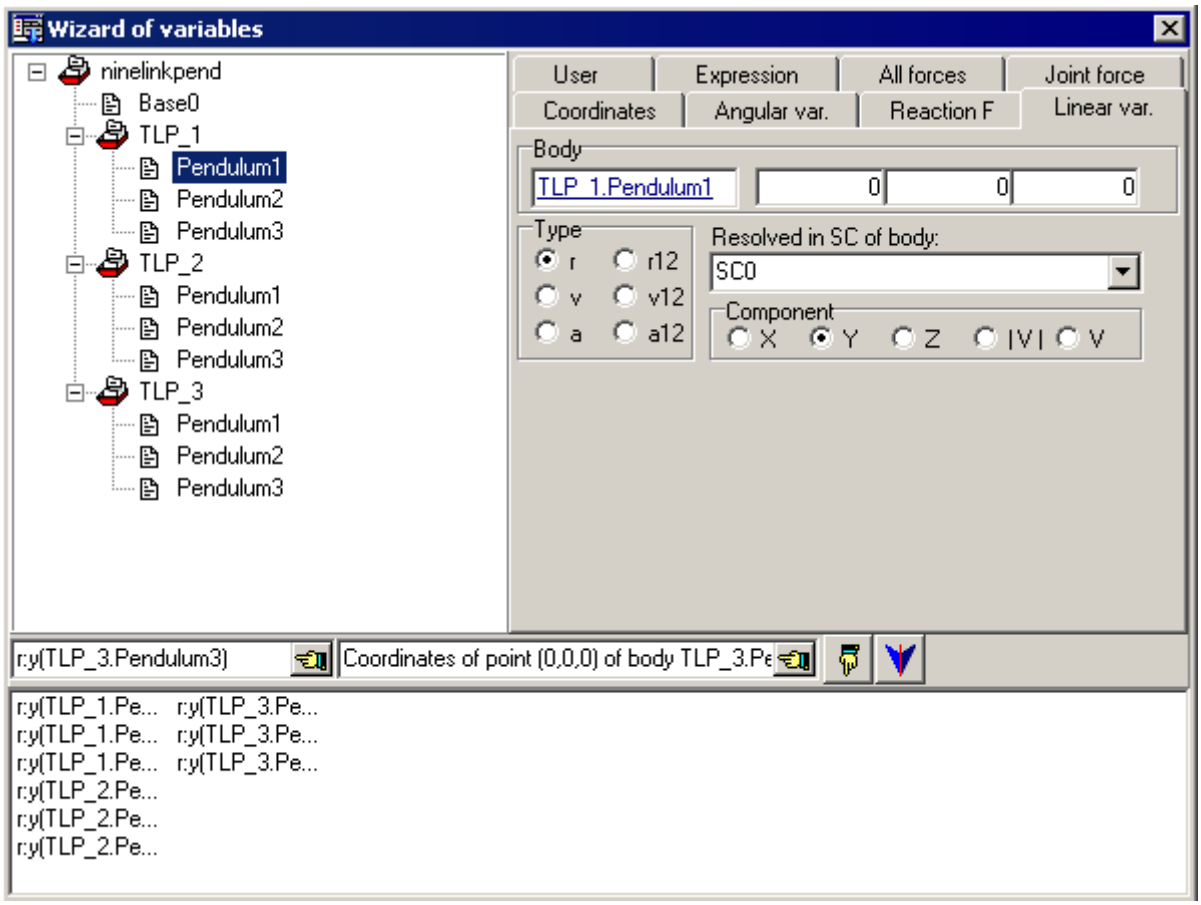
   Pay attention to the ✕ marked strings of the coordinate table. These strings correspond to coordinates of the cut joints (three coordinates of the **jPendulum1** joint in each of the subsystems). Coordinates in cut joints are dependent and are calculated based on values of independent coordinates. Initial values of these coordinates cannot be set by user.



Variables creation

Let us use **Wizard of variables** to create a number of variables – Y coordinate of the center of mass of each of the pendulums.

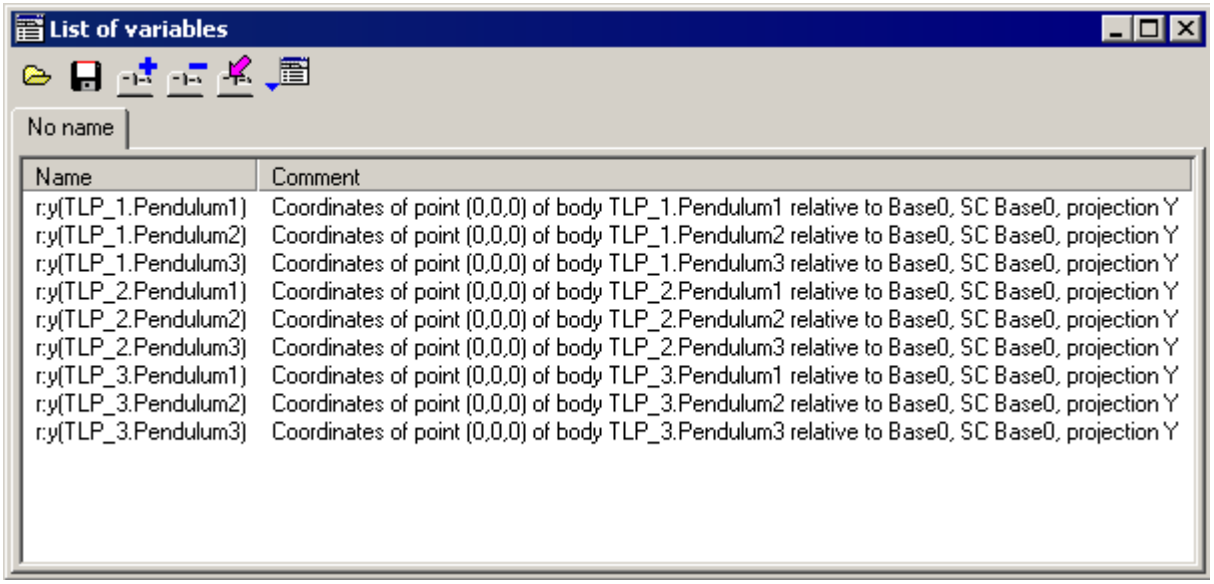
4. Open the **Wizard of variables** (**Tools | Wizard of variables**).
5. Point to the **Linear var.** tab (linear variables: coordinates, velocities, accelerations etc.).
6. Point to the tree of the object elements and select the **Pendulum1** body of the **TLP_1** subsystem.
7. Select **Y** in the **Component** group of the **Linear var.** tab.
8. Then create the new variable to the container with the help of the button . New variable **r:y(TLP_1.Pendulum1)** appears in the container of variables.
9. Add the same variable for each of the rest bodies of the multibody pendulum. The **Wizard of variables** will be look like in the figure below.



Using the list of variables

Let us save the created variables for the further usage with the help of **List of variable** tool.

10. Create new list of variables (**Tools | List of variables**).
11. Select all the variables from the container of the **Wizard of variables** and drag them to the **List of variables** window.


List of variables
No name
Name
Comment
r:y(TLP_1.Pendulum1) Coordinates of point (0,0,0) of body TLP_1.Pendulum1 relative to Base0, SC Base0, projection Y
r:y(TLP_1.Pendulum2) Coordinates of point (0,0,0) of body TLP_1.Pendulum2 relative to Base0, SC Base0, projection Y
r:y(TLP_1.Pendulum3) Coordinates of point (0,0,0) of body TLP_1.Pendulum3 relative to Base0, SC Base0, projection Y
r:y(TLP_2.Pendulum1) Coordinates of point (0,0,0) of body TLP_2.Pendulum1 relative to Base0, SC Base0, projection Y
r:y(TLP_2.Pendulum2) Coordinates of point (0,0,0) of body TLP_2.Pendulum2 relative to Base0, SC Base0, projection Y
r:y(TLP_2.Pendulum3) Coordinates of point (0,0,0) of body TLP_2.Pendulum3 relative to Base0, SC Base0, projection Y
r:y(TLP_3.Pendulum1) Coordinates of point (0,0,0) of body TLP_3.Pendulum1 relative to Base0, SC Base0, projection Y
r:y(TLP_3.Pendulum2) Coordinates of point (0,0,0) of body TLP_3.Pendulum2 relative to Base0, SC Base0, projection Y
r:y(TLP_3.Pendulum3) Coordinates of point (0,0,0) of body TLP_3.Pendulum3 relative to Base0, SC Base0, projection Y

12. Save the list of variables as **ninelinkpend.var** (use the button from the top panel of the window). The directory of the object (**NineLinkPend**) will be used by default for the saving.

Drawing plots

Let us create new graphical window.

13. From the **Tools** menu, select **Graphical window**.
14. Select all the variables from the **List of variables** window and drag them to the graphical window.

Saving results

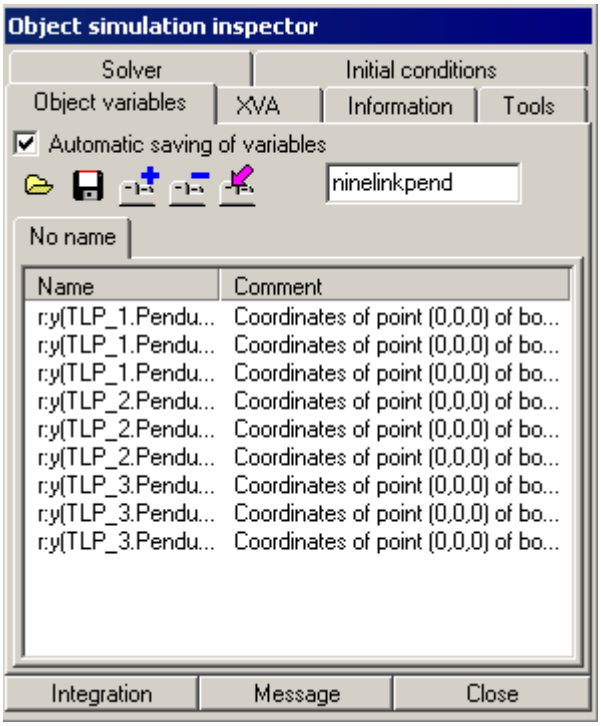
Simulation results (time histories of the variables) can be saved as the file of calculated variables (***.tgr**). From the other hand the user can save plots of the variables in text file with the help of the graphical window tools.

In our example we will use the both approaches. We will compare simulation results for the models created according to the scheme A (Fig.1.7, scheme A) and the scheme B (Fig.1.7, scheme B).

Creation of list of calculated variables

15. Activate the **Object simulation inspector** and point to the **Object variables** tab.
16. Load the list of variables from file **ninelinkpend.var**, or just select all the variables from the **Wizard of variables** container and drag them to the white field of the tab.
17. Turn on the **Automatic saving of variables** box.

Now the variables added to the tab will be automatically saved in the file of calculated variables **ninelinkpend.tgr** during simulation. The directory of the object (**NineLinkPend**) will be used by default for the saving.



Simulation

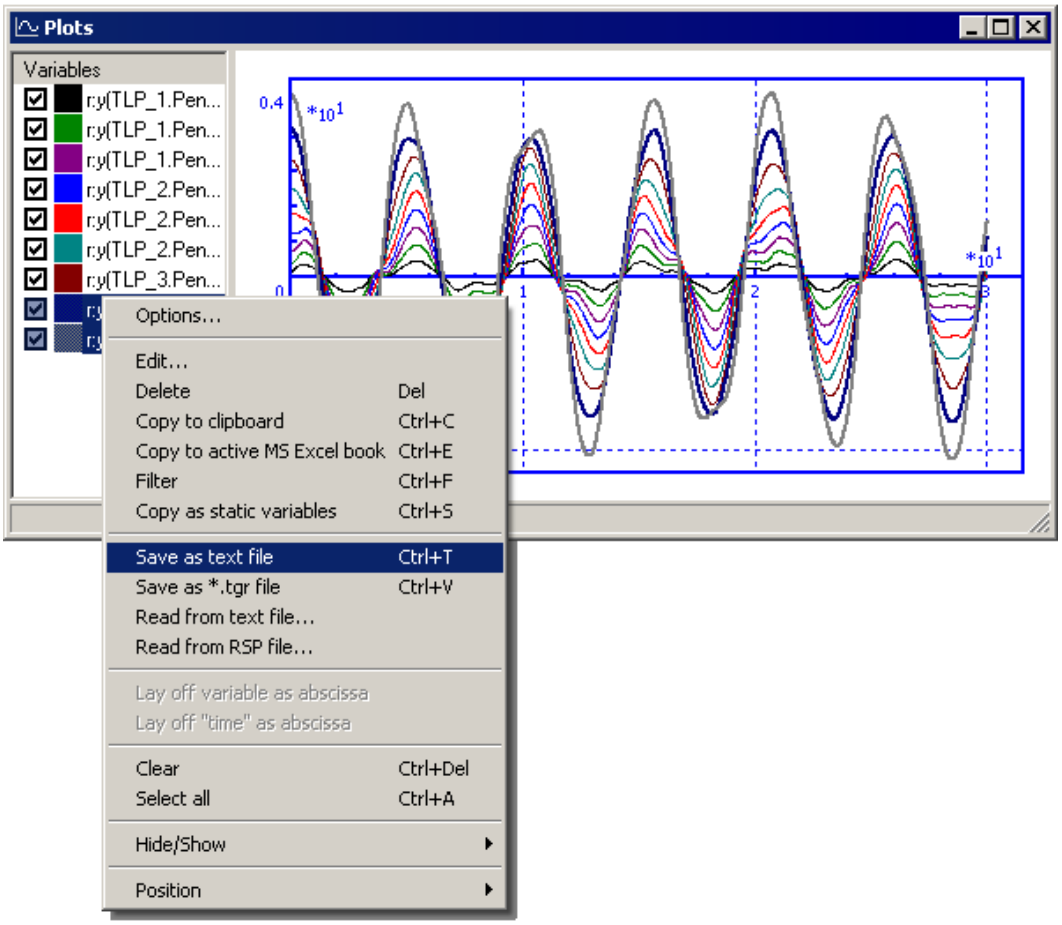
Let us simulate 30 second of motion of the multilink pendulum.

18. Point to the **Solver | Simulation process parameters** tab of the **Object inspector wizard** and set **30** in **Simulation time** field.
19. Click **Integration** button to start simulation. One can minimize animation window to make simulation faster.

You can see the plot of your variables in the graphical window.

Saving plots in text files

20. Point to the graphical window. Then select the two last variables in the list of variables of the window and click on the right mouse bottom. The popup menu will be opened. Select the **Save as text file** tool from the menu and save the plots to **result.txt** text file. The directory of the object (**NineLinkPend**) will be used by default for the saving.



Saved plots can be loaded to any graphical window as *static variables*. Plots of static variables will not be calculated again during simulation.

1.5.2.3. Connections usage

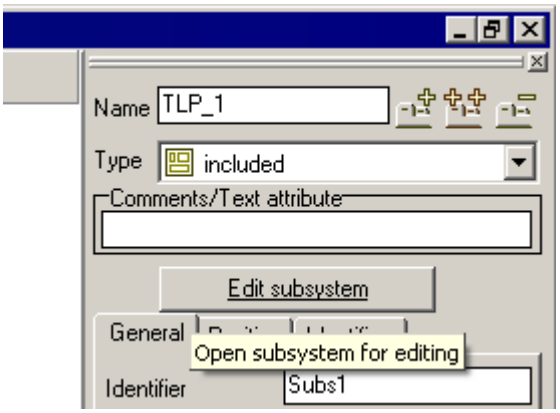
Previously we considered joints between the bodies of different subsystems of the compound model as the elements of the compound object (Fig. 1.7, scheme A, page. 33).

Now we will use another approach. We will describe kinematics of the multi-body pendulum model with the help of *connections* (Fig. 1.7, scheme B, page 33). Thereto we will make some changes in the subsystem models as well as the compound model **NineLinkPend**.

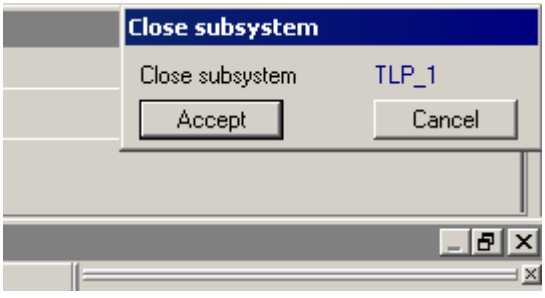
Then we will simulate motion of the pendulum and compare the results with the previously saved ones.

Internal subsystem editing

1. Close simulation program **UM Simulation** and come back to the **UM Input** program.
2. Point to the **Subsystems** element of the **Ninelinkpend** object elements tree. Select the **TLP_2** subsystem.



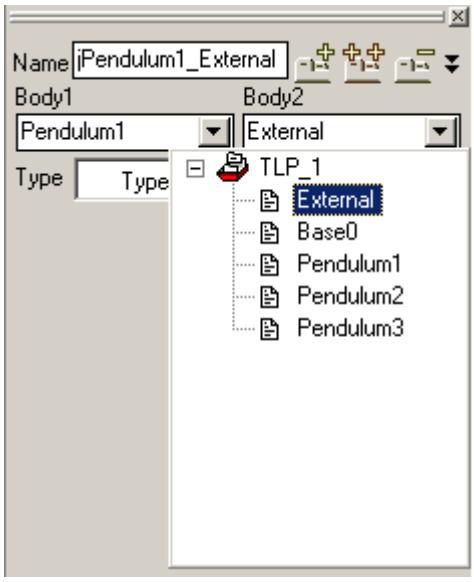
3. Click **Edit subsystem** button.
   TLP_2 subsystem object will be opened in the UM Input program window. One can edit the internal subsystem as an ordinary UM object, but the correspondent compound object is not available during this session. Pay attention to the **Close subsystem** window in the top left corner of the screen. After some modifications in the subsystem the user can confirm or cancel all changes in it.



Adding the joints with the External body

Let us add to the subsystem **TLP_2** the joint between the **Pendulum1** and the **External** bodies.

4. Point to the **Joints** element, create new joint and rename it as **jPendulum1_External**.
5. Select the **Pendulum1** body in the **Body1** drop-down list and the **External** body in the **Body2** list as it is shown in the figure below.
6. Change its **Type** to **Rotational**.
7. Set **1** for the **Z** coordinate of the **Body1**.



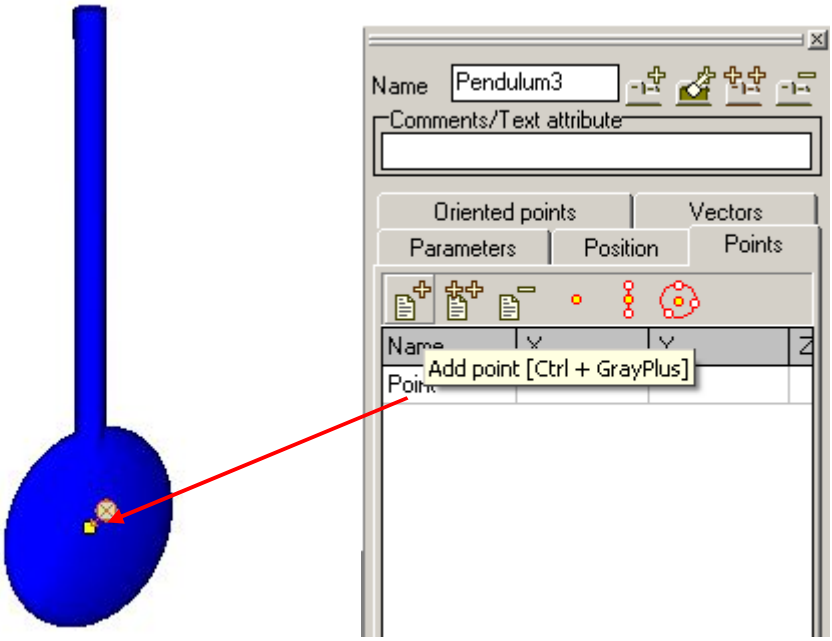
8. Copy the **jPendulum1_External** to the clipboard (main menu **Edit | Copy to clipboard**, see Fig.1.9 page. 38).
9. Click the **Accept** button of the **Close subsystem** window. The **TLP_2** subsystem will be closed, **NineLinkPend** model will be activated.
10. Start edit of the **TLP_3** subsystem (select the subsystem from the compound object and click the **Edit subsystem** button).
11. Paste the previously saved joint from the clipboard (main menu **Edit | Paste**). The new element will appear on the **Joints** element of the subsystem elements tree.
12. Point the new joint and select the **Pendulum1** body as the **Body1**.
13. Click the **Accept** button of the **Close subsystem** window. The **TLP_3** subsystem will be closed, **NineLinkPend** model will be activated.

Adding the joints with the **External** body is finished.

Connection points adding

Connection points are necessary for the description of the *connections*. We should add the points to the **TLP_1.Pendulum3** and the **TLP_2.Pendulum3** bodies to describe the joints between the subsystems.

14. Start edit of the **TLP_3** subsystem (select the subsystem from the compound object elements tree and click the **Edit subsystem** button).
15. Point to the **Bodies** element of the subsystem object elements tree and select the **Pendulum3** body.
16. Come to the **Points** tab for the **Pendulum3** and add a new point (click the button). A new point with zero coordinates in coordinate frame of the **Pendulum3** body will be added by default. Coordinates of the point are shown in the table (empty fields correspond to zero values). Position of the point is marked on the image of the body. The position of the point coincides with the position of the rotational axis between two pendulums.

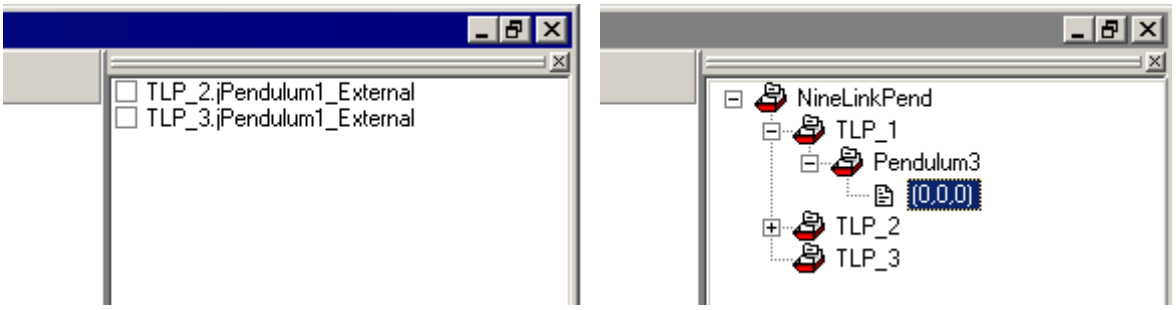


17. Click the **Accept** button of the **Close subsystem** window. The **TLP_1** subsystem will be closed, **NineLinkPend** model will be activated.
18. Add the connection point for the **Pendulum3** body of the **TLP_2** subsystem.
19. Click the **Accept** button of the **Close subsystem** window. The **TLP_2** subsystem will be closed, **NineLinkPend** model will be activated.

Subsystem editing is finished.

Connections description

20. Point to the **Connections** element of the **NineLinkPend** object elements tree. The element **Connections** is the list which contains all joints and force elements with the second body that is external. In our case the list includes **jPendulum1_External** joints of **TLP_2** and **TLP_3** subsystems.



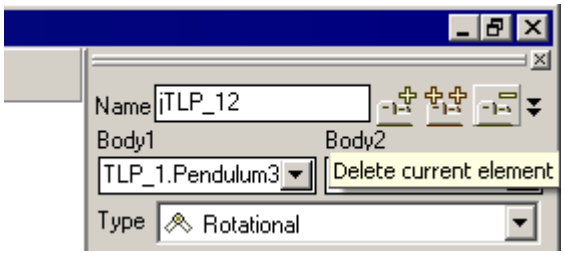
21. Double click on the box on the left of the element **TLP_2.jPendulum1_External** in the list. Object tree will be opened where the lower level is the level of connection points.
22. Select the point **(0, 0, 0)** of the **Pendulum3** body of the **TLP_1** subsystem. The connection is ready.
23. Double click on the box on the left of the element **TLP_3.jPendulum1_External** in the list.
24. Select the point **(0, 0, 0)** of the **Pendulum3** body of the **TLP_2** subsystem. The connection is ready.

Description of the connections is completed.

Deleting of the redundant joints

The earlier created joints **jTLP_12** and **jTLP_23** of the compound object duplicate the joints described with the **External** body. Let us delete them.

25. Point to the **Joints** element of the **NineLinkPend** object elements tree.
26. Delete the **jTLP_12** and **jTLP_23** joints.

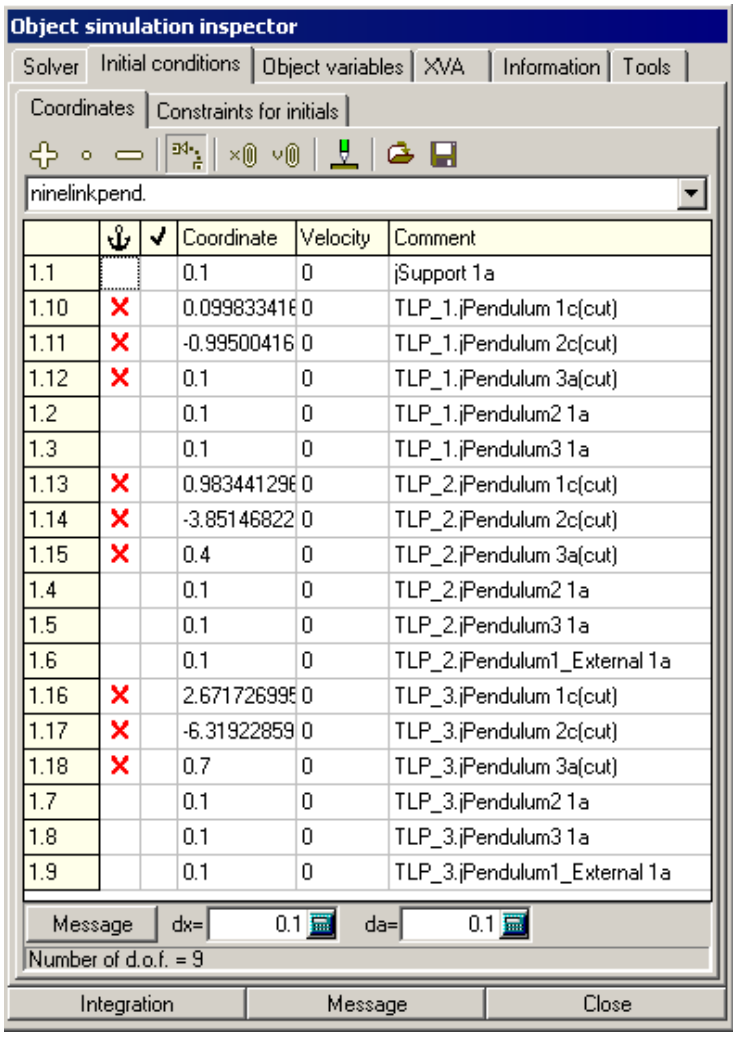


Comparison of results

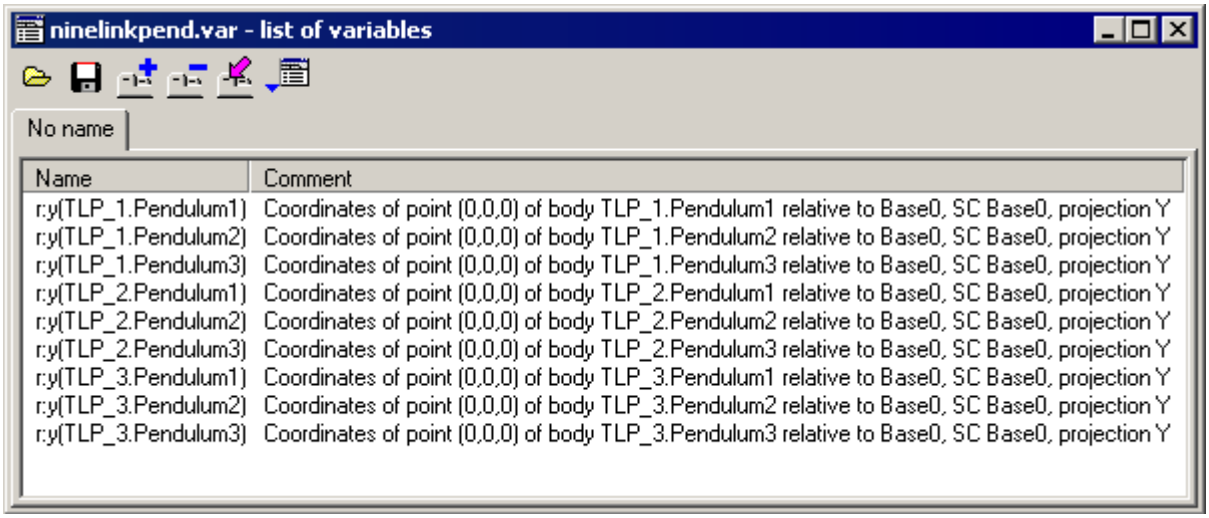
Let us compare the simulation results with the previously saved data and make sure that the different variants of description of joints between subsystems (fig. 1.7, scheme A and scheme B) give identical results.

We will plot the Y displacement of the last two pendulums and compare them with the data from the file of calculated variables **ninelinkpend.tgr** and the plots saved in the text file **results.txt**.

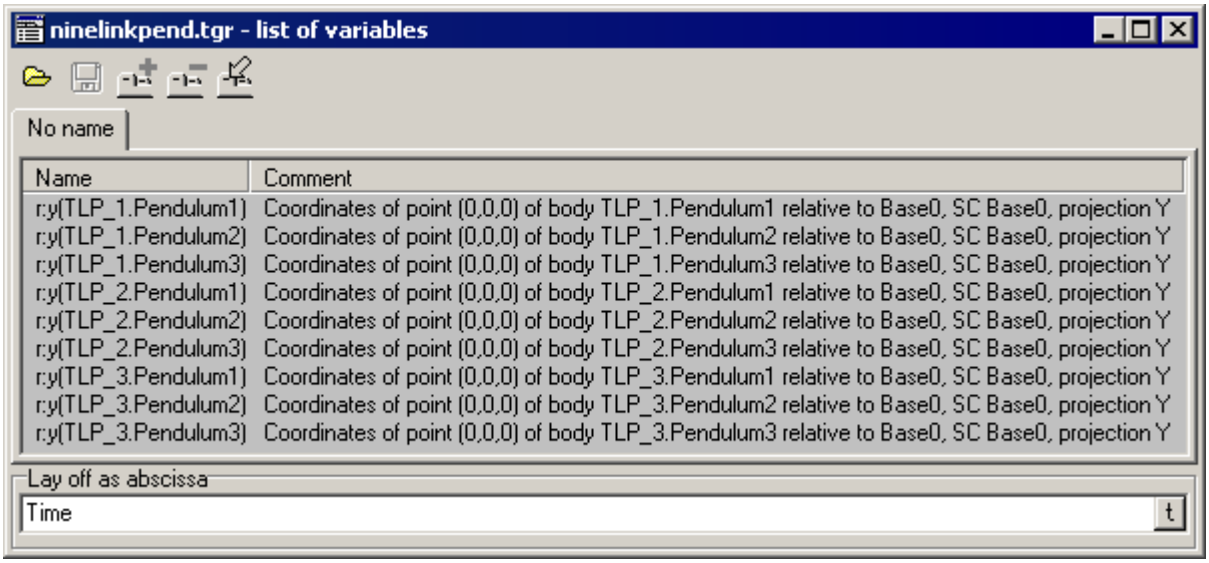
27. Select **Object/Simulation** menu item. **UM Simulation** program starts and loads the current model. If autosave of configuration option is enabled, previously created animation and graphical windows will be opened. One can use **Autosave** tab of the **Options** window (main menu **Tools | Options**) to enable autosave options.
28. From the **Analysis** menu, select **Simulation**. **Object simulation inspector** appears.
29. Point to the **Initial conditions | Coordinates** tab. If autosave of initial conditions option is enabled, previously values will be used. Set **0.1** value for all the uncut joints as it is shown in the figure below if not.



30. Create a new graphical window (**Tools | Graphical window**).
31. Create a new **List of variables…** window (**Tools | List of variables…**) and load file **ninelinkpend.var**.


ninelinkpend.var - list of variables
No name
Name
Comment
r:y(TLP_1.Pendulum1) Coordinates of point (0,0,0) of body TLP_1.Pendulum1 relative to Base0, SC Base0, projection Y
r:y(TLP_1.Pendulum2) Coordinates of point (0,0,0) of body TLP_1.Pendulum2 relative to Base0, SC Base0, projection Y
r:y(TLP_1.Pendulum3) Coordinates of point (0,0,0) of body TLP_1.Pendulum3 relative to Base0, SC Base0, projection Y
r:y(TLP_2.Pendulum1) Coordinates of point (0,0,0) of body TLP_2.Pendulum1 relative to Base0, SC Base0, projection Y
r:y(TLP_2.Pendulum2) Coordinates of point (0,0,0) of body TLP_2.Pendulum2 relative to Base0, SC Base0, projection Y
r:y(TLP_2.Pendulum3) Coordinates of point (0,0,0) of body TLP_2.Pendulum3 relative to Base0, SC Base0, projection Y
r:y(TLP_3.Pendulum1) Coordinates of point (0,0,0) of body TLP_3.Pendulum1 relative to Base0, SC Base0, projection Y
r:y(TLP_3.Pendulum2) Coordinates of point (0,0,0) of body TLP_3.Pendulum2 relative to Base0, SC Base0, projection Y
r:y(TLP_3.Pendulum3) Coordinates of point (0,0,0) of body TLP_3.Pendulum3 relative to Base0, SC Base0, projection Y

32. Drag the two last variables from the **ninelinkpend.var – list of variables** window to the new graphical window. The plots of the variables will be calculated during simulation.
33. Use **Tools | List of calculated variables…** to load the file of calculated variables **ninelinkpend.tgr**.

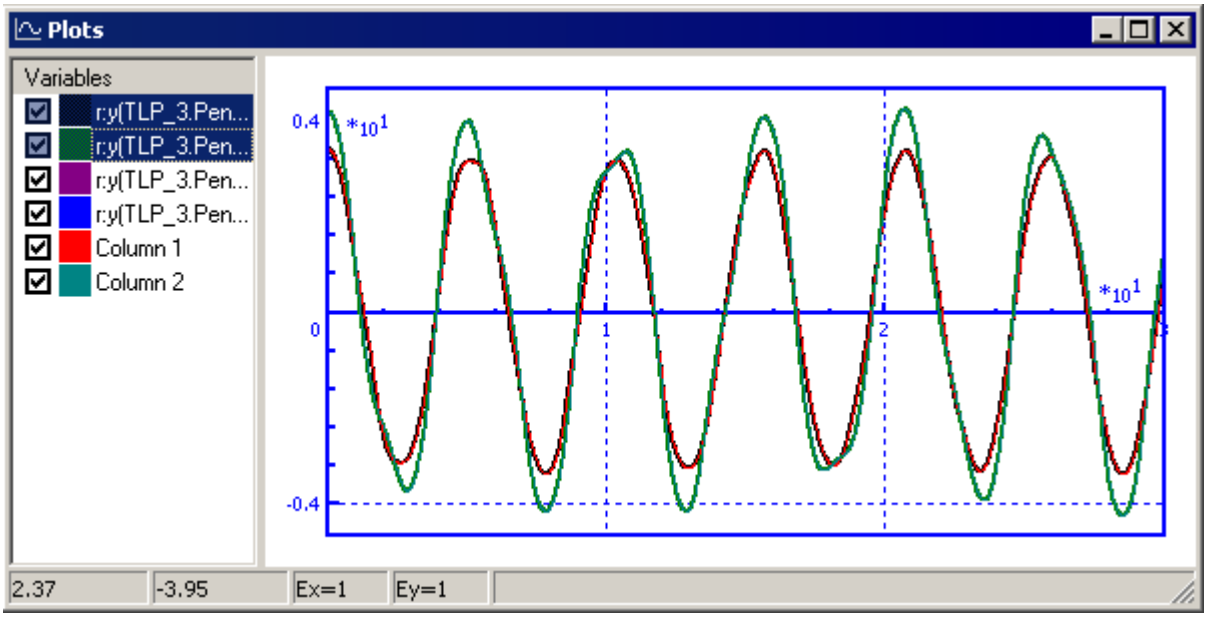

ninelinkpend.tgr - list of variables
No name
Name
Comment
r:y(TLP_1.Pendulum1) Coordinates of point (0,0,0) of body TLP_1.Pendulum1 relative to Base0, SC Base0, projection Y
r:y(TLP_1.Pendulum2) Coordinates of point (0,0,0) of body TLP_1.Pendulum2 relative to Base0, SC Base0, projection Y
r:y(TLP_1.Pendulum3) Coordinates of point (0,0,0) of body TLP_1.Pendulum3 relative to Base0, SC Base0, projection Y
r:y(TLP_2.Pendulum1) Coordinates of point (0,0,0) of body TLP_2.Pendulum1 relative to Base0, SC Base0, projection Y
r:y(TLP_2.Pendulum2) Coordinates of point (0,0,0) of body TLP_2.Pendulum2 relative to Base0, SC Base0, projection Y
r:y(TLP_2.Pendulum3) Coordinates of point (0,0,0) of body TLP_2.Pendulum3 relative to Base0, SC Base0, projection Y
r:y(TLP_3.Pendulum1) Coordinates of point (0,0,0) of body TLP_3.Pendulum1 relative to Base0, SC Base0, projection Y
r:y(TLP_3.Pendulum2) Coordinates of point (0,0,0) of body TLP_3.Pendulum2 relative to Base0, SC Base0, projection Y
r:y(TLP_3.Pendulum3) Coordinates of point (0,0,0) of body TLP_3.Pendulum3 relative to Base0, SC Base0, projection Y
Lay off as abscissa
Time
t

34. Drag the two last variables from the **ninelinkpend.tgr – list of variables** window to the graphical window. Static plots of the variables will appear in the window.
35. Start simulation.
36. Point to the graphical window and make right mouse click on its **Variable list** panel. Use the **Read from text file…** item of the popup menu to load data from the **results.txt** file.

Three pairs of plots corresponded to displacements of centers of mass of the two last pendulums are displayed in the graphical window now.

One can see that evaluated results (first pair of plots) and saved data (second and third pairs of plots) completely coincide.

The plot of the variables selected in the list of graphical window is marked with the line of double width. Use this feature to make sure that the all three pairs of plots from the list are equal.



2. Free and forced oscillations

2.1. What we will learn

In this lesson we will learn how to add forces, preset movement of a body as a time function and use parameterization of a model. We will use **Linear analysis** for obtaining the equilibrium position of a system, natural frequencies and forms. As well as we will analyze the spectrum of output data using the **Statistics** tool.

2.2. Model scheme

The example of simulation of free and forced damped oscillations is considered. In this lesson we will create the model shown in the Fig. 2.1. Model consists of two rigid bodies **Top** and **Brick**, two translational joints, a linear spring and a damper. We will set vertical coordinate of the upper body as a sinusoid function.

You can find the final model in the **{um_root}\samples\tutorial\oscillator** directory or download it using the following link:
http://www.umlab.ru/download/50/oscillator.zip

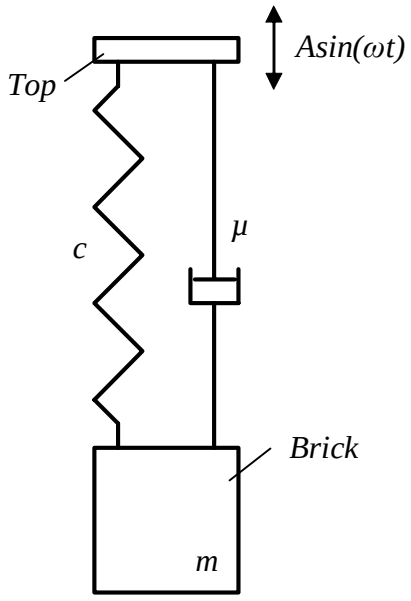


Figure 2.1. Model scheme

2.3. Creating the model

2.3.1. Running UM Input and creating new model

Running UM Input program

1. Click **Start/Programs/Universal Mechanism 5.0/UM Input**.

Creating a new model

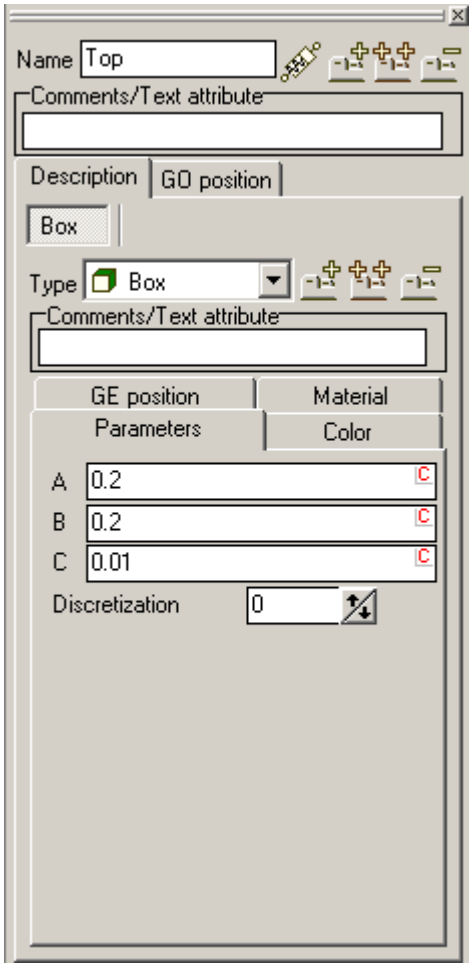
1. From the **File** menu point to **New object**. The window of the constructor appears.

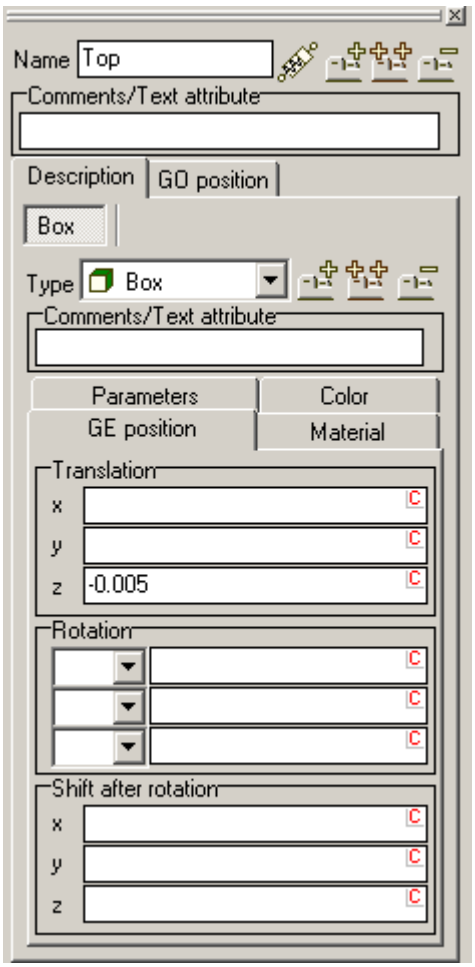
2.3.2. Creating graphical objects

Top

We will create a thin rectangular plate as a graphical object for the **Top** body.

1. Create new graphical object.
2. Set its name to **Top**.
3. Add new graphical element – **Box**.
4. Set parameters and GE position for the **Box** as it is shown below.
5. Select the **Color** tab and choose **blue** for **diffuse** and **specular** colors.

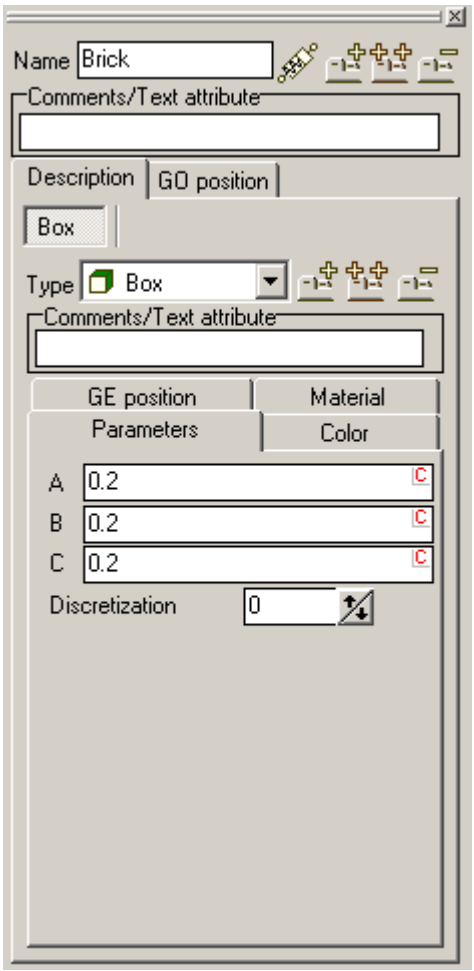


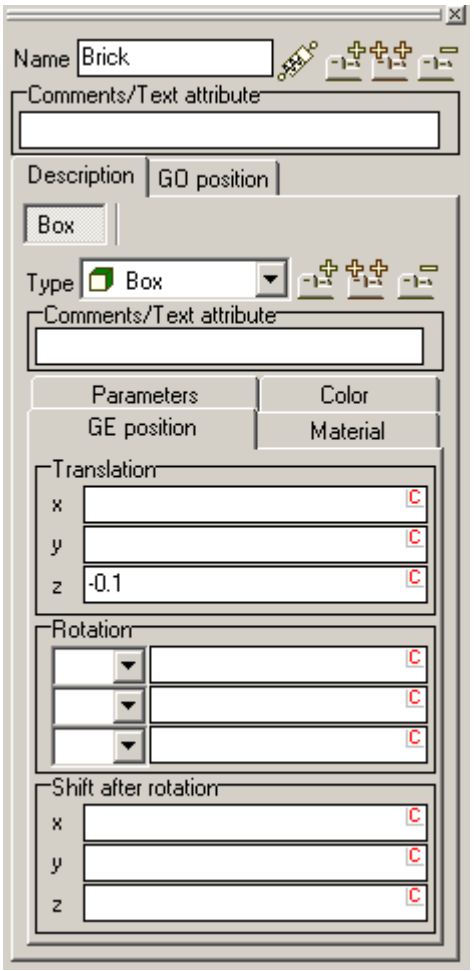


Brick

Let us describe the brick as a cube of **0.2 m** side length.

1. Create new graphical object.
2. Set its name to **Brick**.
3. Add new graphical element **Box** into this graphical object.
4. Set its parameters and GE position.
5. Select the **Color** tab and set **red** for **diffuse** and **specular** colors.

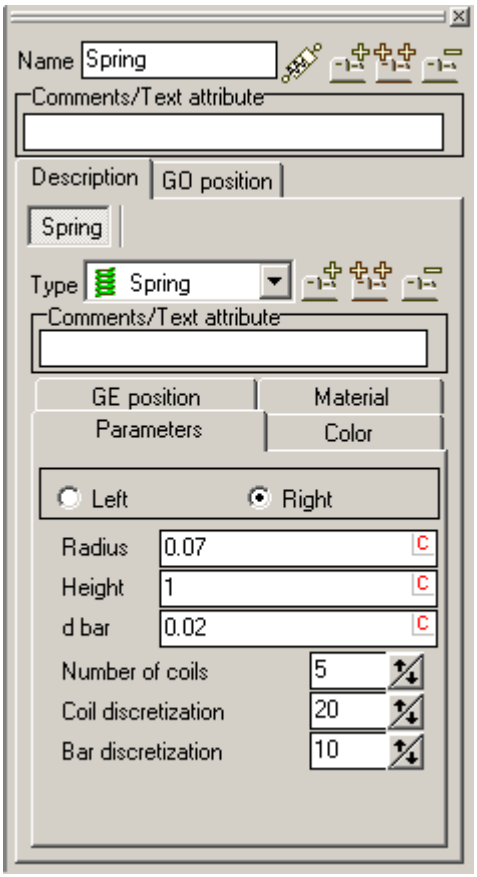




Spring

Now we will create the graphical object for the spring.

1. Create new graphical object.
2. Set its name to **Spring**.
3. Add new graphical element – **Spring**.
4. Set **Spring** parameters as it is shown below.

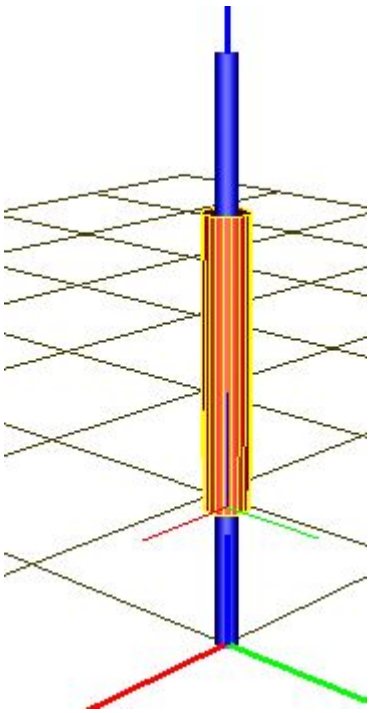


Damper

Now we come to the last graphical object in this model – damper.

1. Create new graphical object.
2. Set its name to **Damper**.
3. Add new graphical element for the **Damper** – **Cone**.
4. Set parameters as follows:

   R2 = 0.02;
   R1 = 0.02;
   h = 1.
5. Select the **Color** tab and choose **blue** for **diffuse** and **specular** colors.
6. Add one more **Cone** with the following parameters: **R2 = 0.04**; **R1 = 0.04**; **h = 0.5**. In the **GE Position** tab set **Translation/Z** to **0.25**. Select **red** **diffuse** and **specular** colors.

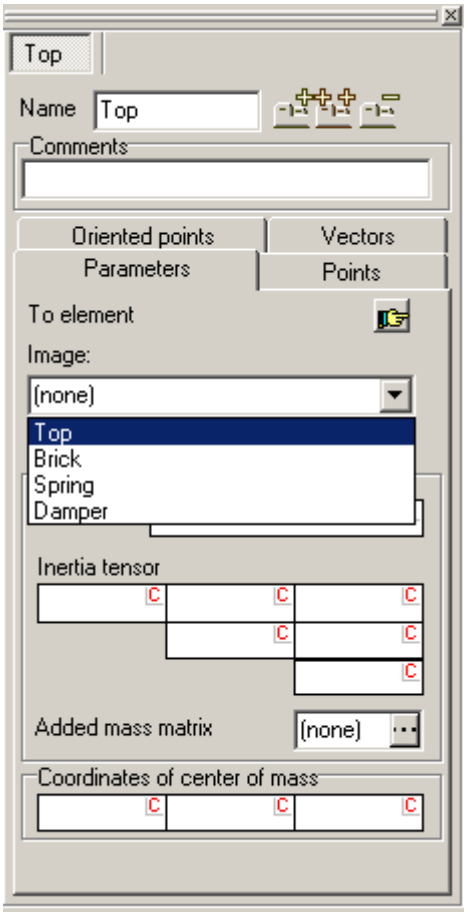


2.3.3. Creating rigid bodies

Top

Create new rigid body – **Top**.

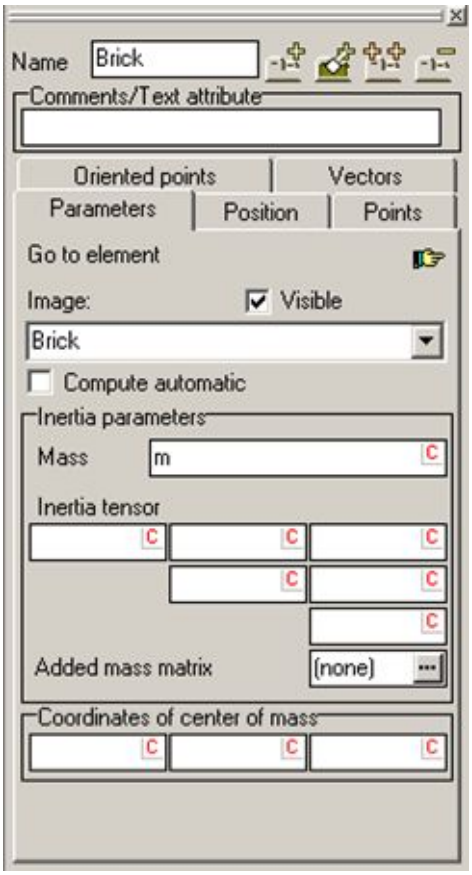
1. Add new rigid body.
2. Set its name to **Top**.
3. In the **Image** list select **Top**.
4. Leave **Mass** and **Inertia tensor** empty.

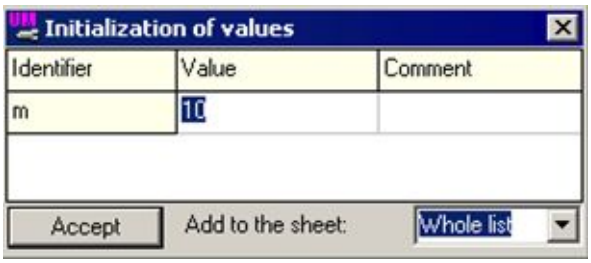


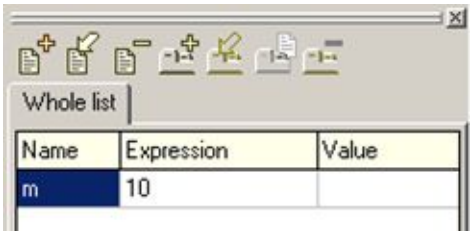
Brick

Now we will create one more rigid body – **Brick**. Its mass we will express via parameter (identifier) **m**. Such a parameterization gives us a possibility to change its mass easily and quickly obtain results for various values of the mass of the brick without regeneration equations of motion. Otherwise we would have to generate equations every time we want to change its mass.

1. Add new rigid body.
2. Rename it to **Brick**.
3. In the **Image** list select **Brick**.
4. Set **Mass** to **m** and press **Enter**. New **Initialization of values** window appears.
5. Set **Value** to **10**. Press **Enter**.
6. This new parameter appears in the parameter list in the bottom left corner of the constructor window, see the figure below.







2.3.4. Creating joints

Joint for the top

The **Top** body moves along the vertical direction according $A \cdot sin(\omega \cdot t)$ function. Now we will describe the translational joint between the base and the **top** and set the coordinate in this joint as a time function.

1. Select the **Top** body in the tree of elements.
2. Click the button. Select **Create joint** and in the drop-down list select **Translational**, see the Fig. 2.2, left. The new joint of this type is created. Now you can see parameters of the joint.
3. Select the **Geometry** tab and set joint parameters as it is shown in the Fig. 2.2, right.

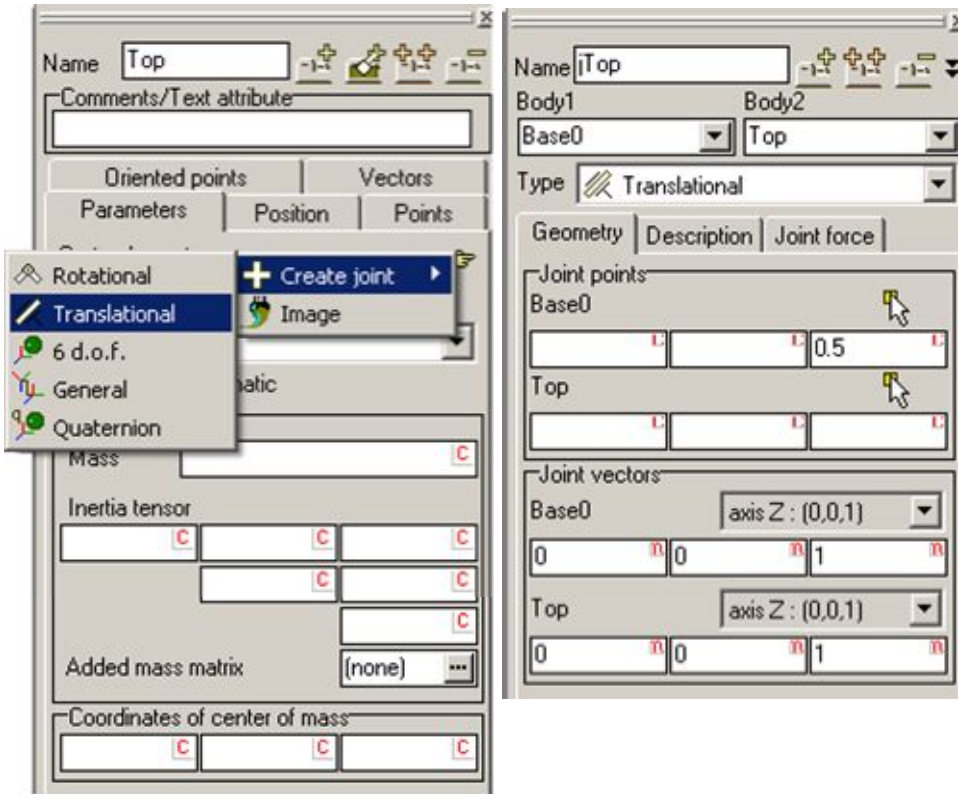


Figure 2.2. Creating translational joint

4. Point to **Description** tab.
5. Turn on the **Prescribed function of time** check box.
6. Set the **Type of description** to **Expression**, and then input **a*sin(omega*t)**, see the Fig. 2.3, and press **Enter**.

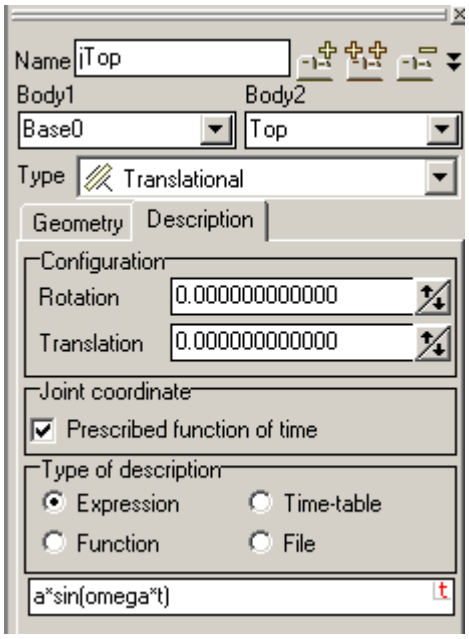
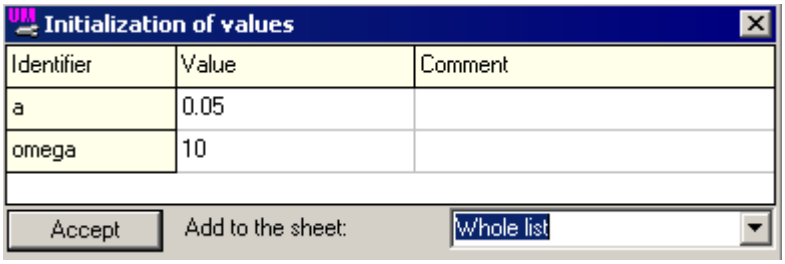


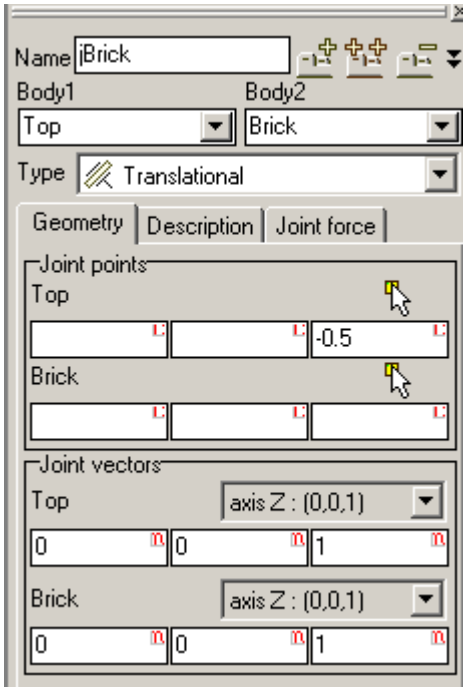
Figure 2.3. Prescribed function of time

7. In the **Initialization of values** window set **a = 0.05** (m) and **omega = 10** (rad/s).



Joint for the brick

1. Select the **Brick** body in the tree of elements.
2. Click the button.
3. In the drop-down list select **Translational** again.
4. Select the **Top** as the first body instead **Base0**, see the figure below.
5. Set the rest parameters of the joint as it is shown below.



2.3.5. Creating force elements

Now we will describe elastic and damping force elements between the top and the brick. Let us use **c** parameter for the stiffness coefficient of the spring and **mu** parameters for the damping coefficient of the damper. Length of the unloaded spring let us denote as **l0**.

1. Select the **jBrick** joint.
2. Select the **Joint force** tab.
3. In the **Joint force** list select the **Linear**.
4. In the **c** box input **c** (stiffness coefficient), in the **x0** box input **l0** and set **d** to **mu**, see Fig. 2.4. Press **Enter**. Set values of parameters as follows: **c = 250**, **l0 = 0.4**, **mu=5**.

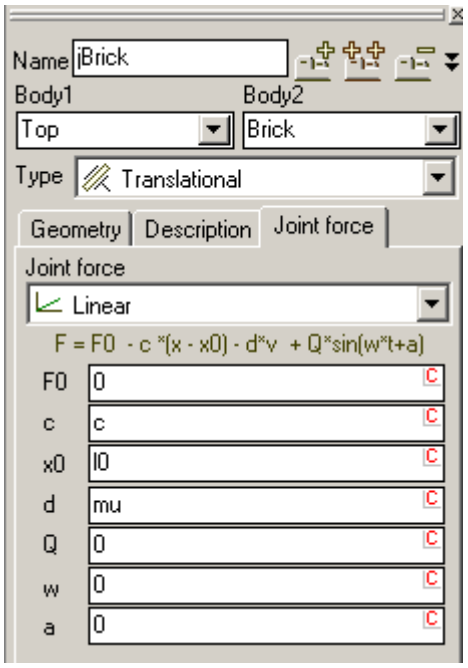


Figure 2.4. Elastic and damping joint forces

2.3.6. Visualization of spring and damper

After all we have completely described object from the mechanical point of view. We described all elements we need: rigid bodies, joints and force elements. However our model now looks not so good – spring and damper introduced as joint forces that cannot be visualized, see the Fig. 2.5, left. In order to visualize spring and damper we will create two **bipolar** forces in the model. Their values we set to zero. That is why these bipolar forces will not influence on the dynamics of the model, but give us a possibility to show the spring and the damper, see the Fig. 2.5, right.

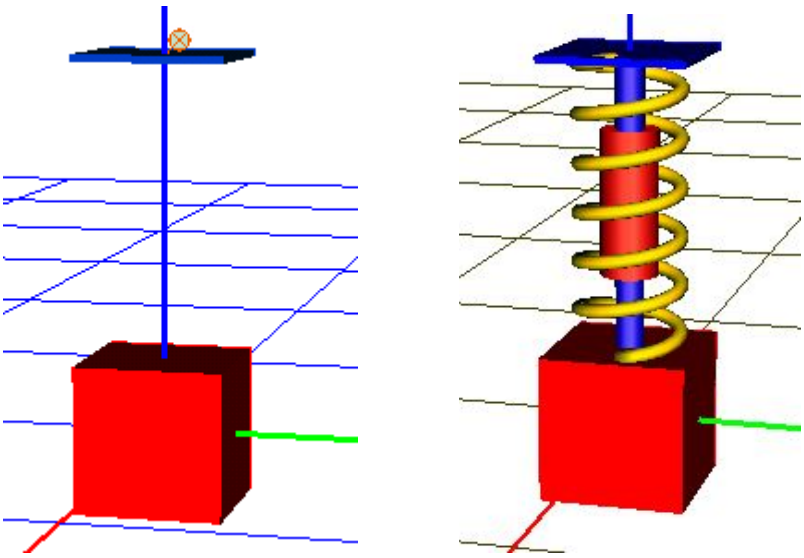


Figure 2.5. Visualization of forces

Note. There are several possible ways to describe elastic and damping forces in our model. We used the way to describe them as joint forces, but it is not the only right way. We could introduce them as bipolar as well. And in this latter case we would visualize them and introduce forces at once without intricate describing additional fake bipolar forces.

But such a way leads to a following problem. Our ideal case that we consider here allows to model the situation when the length of the spring and damper equal to zero. Imagine that the brick has so large amplitude that the attachment points of the spring and damper will be on the same level. In such a case we have degeneration of bipolar forces that act along the line between the attachment points. When we have zero length we could not find the direction of the bipolar forces. Joint forces have no such a degeneration, because they direction always coincide with the axis of the joint. That is why we used very joint forces here.

1. So, select the **Bipolar force** in the tree of elements.
2. Add two bipolar force elements. Set their parameters as it is shown in the Fig. 2.6.

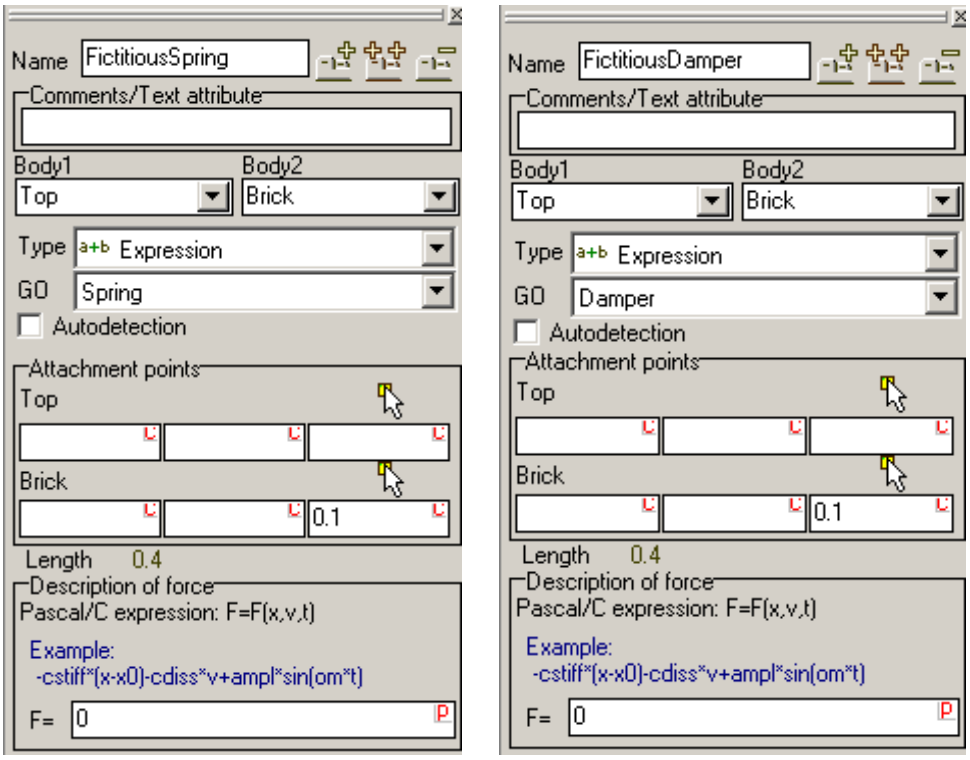


Figure 2.6. Fictitious bipolar forces

2.3.7. Additional parameters

Using UM you can express one parameter via others. Here we will add two new parameters in the model – accurate analytical values of the natural frequency and the critical damping coefficient. Our model is very simple that is why we can obtain analytical solutions easily.

Natural frequency can be obtained according to the following formula:

$$k = \sqrt{\frac{c}{m}}, \text{ where}$$

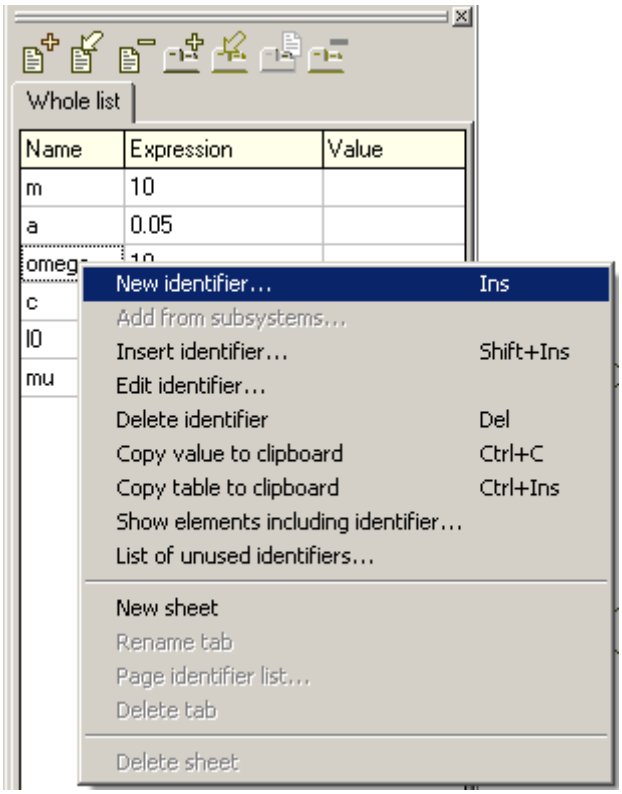
k – natural frequency, rad/s;
c – stiffness coefficient, N/m;
m – mass of the body, kg.

Critical damping coefficient can be found as:

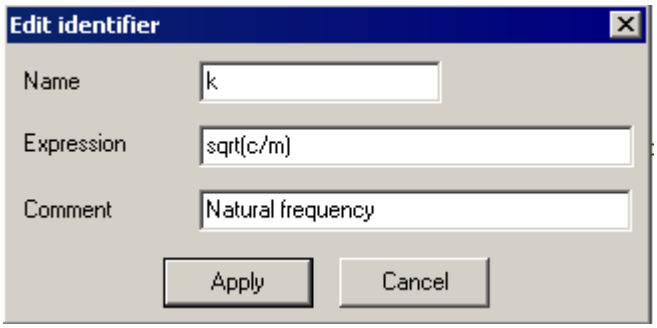
$$\mu^* = 2\sqrt{cm}, \text{ where}$$

μ^* – critical damping coefficient, Ns/m.

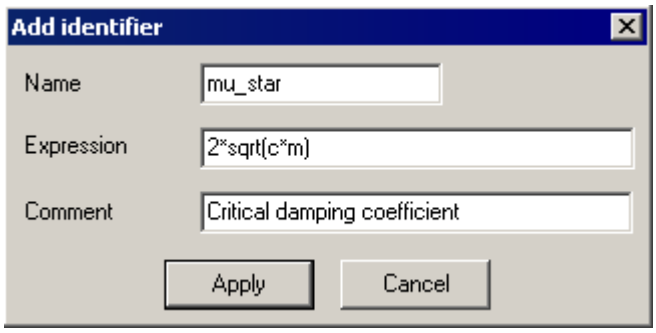
3. Well, add new identifiers (parameters) to our model. Click the button in the list parameters or select the **New identifier** menu command from the context menu.



4. Fill out the **Add identifier** form as it is shown in the figure below.



5. Add one more identifier: **mu_star = 2*sqrt(c*m)**. It is a critical damping coefficient.



2.3.8. Preparation for simulation

1. In the tree of elements select **Object**.
2. Set **Generation of equations** to **Numeric-iterative** (see **Inspector** window in the right part of the constructor window).
3. Save the model as **Oscillator** (use menu command **File/Save as**).

   Now we will come to the simulation program.

4. From the **Object** menu select **Simulation**, or simply press **F9** key.

   The simulation programs starts and opens the current model.

2.4. Simulation of motion

Let us consider some particular cases of oscillations: free damped oscillations and forced oscillations without damping.

2.4.1. Free oscillations

Free damped oscillations

1. Open new animation window (**Tools/Animation window**).

Open new graphical window, where we will plot time history of the vertical position of the brick.

2. Open new graphical window (**Tools/Graphical window**).
3. Open **Wizard of variables** (**Tools/Wizard of variables**).
4. Select the **Linear var.** (linear variables) tab, select **Brick** in the list of bodies, set **Type** to **r** (coordinate), set **Component** to **Z**. Click the button to create new variable. The variable appears in the container of variables. Drag the variable to the graphical window. Close the **Wizard of variables**.
5. From menu **Analysis** select **Simulation**. The **Object simulation inspector** appears.
6. Arrange windows on the desktop as you prefer, for example, as it is shown in the Fig. 2.7.

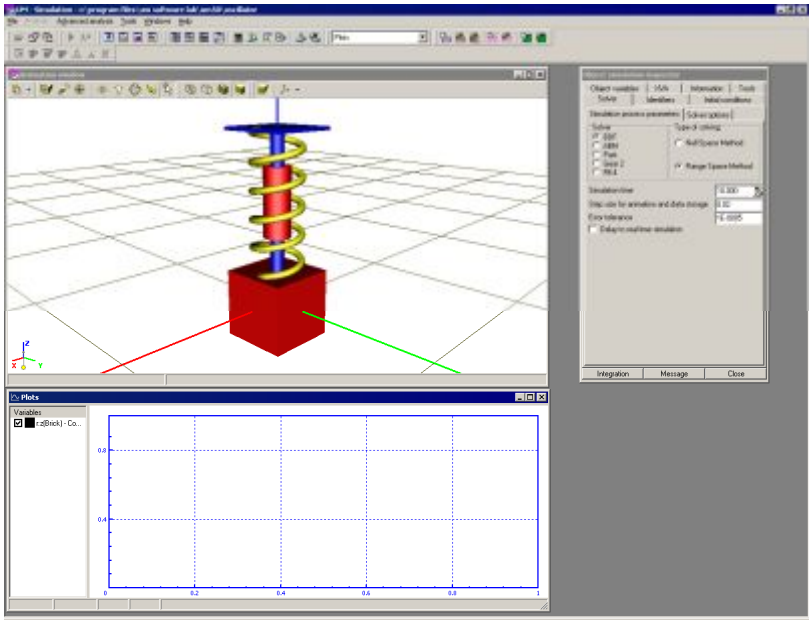


Figure 2.7. Desktop of the simulation program

7. Select the **Object simulation inspector** and click the **Identifiers** tab.
8. Set **a** to **0** and press **Enter**. So we set zero amplitude of the oscillations of the **Top** body, in other words we fix the **Top** in order to analyze free oscillations.

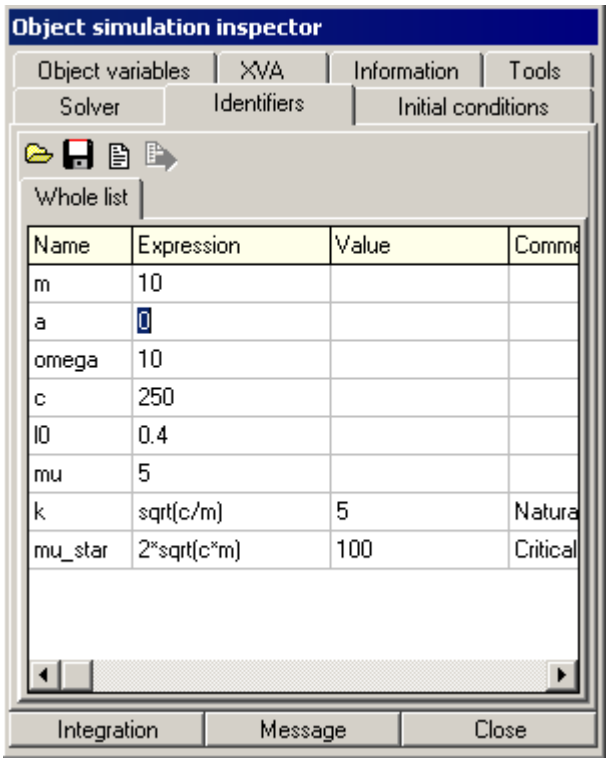
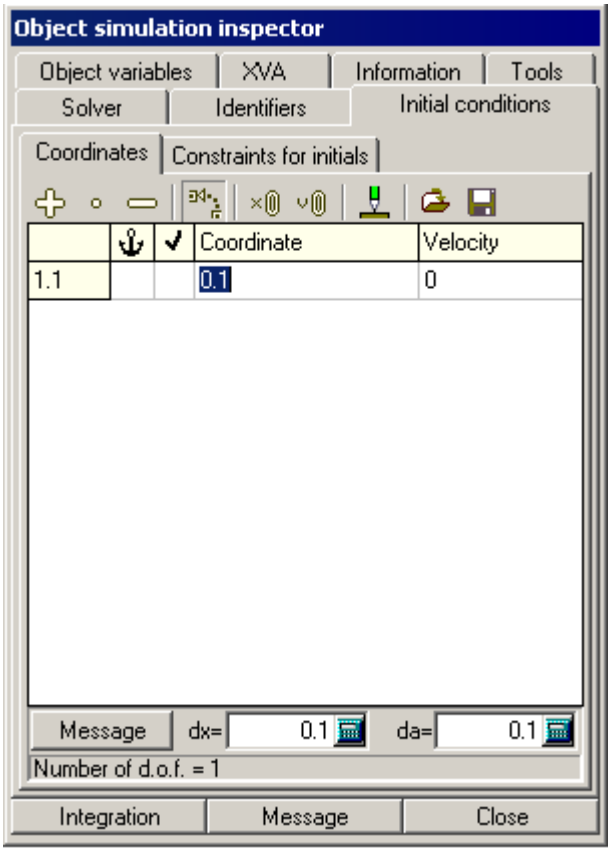


Figure 2.8. Parameters of the model

9. Select the **Initial conditions** tab. In the **Coordinate/1.1** input **0.1**. We need to shift the brick a bit because its position at zero coordinate is quite near to its equilibrium position that gives us small amplitude of oscillations if we do not shift the body.



10. Select the **Solver** tab. Set **Simulation time** to **25** (seconds).
11. Run simulation clicking the **Integration** button.

Process of the numerical simulation starts for 25 seconds period. You can see oscillations of the **Brick** in the animation window and time history of the vertical position of the brick.

12. Click the **100%** button in the drop-down tool panel in the top or click the **Show all** menu command in the context menu, see the Fig. 2.9. Plot now fits the window.

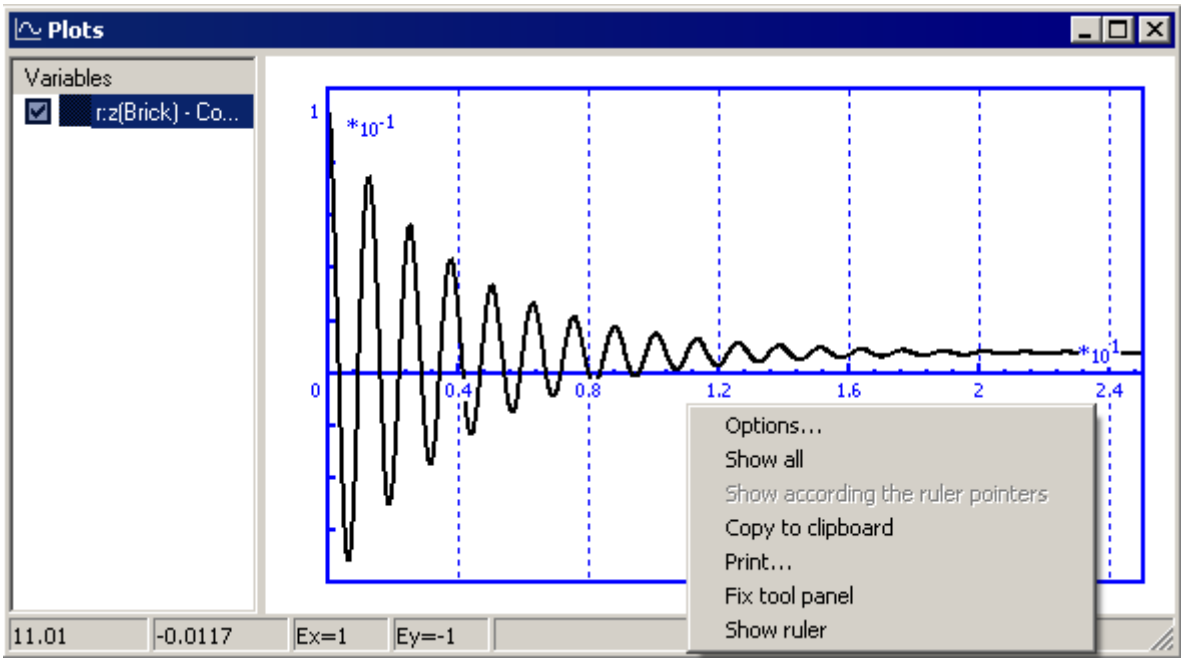


Figure 2.9. Graphical windows after the first experiment

Free oscillations without damping

Now we will turn off damping and compare plots for damped and free oscillations. Using zero damping coefficient gives us free oscillations.

1. Select the graphical window. Point to the **r:z(Brick)** variable in the list of variables. Open context menu. Select the **Copy as static variable** menu command. The second variable appears.
2. Select the **Pause** inspector and click the **Interrupt** button. **Object simulation inspector** appears.

Note. The **r:z(Brick)** variable, which we dragged from the **Wizard of variables**, will be recalculated for every numerical experiment. It is so-called *dynamic* variable. In order to compare plots for different experiments we need to copy *dynamic* variables as *static* ones. *Static* variables are not changed from one experiment to another.

3. Select the **Object simulation inspector** and point to the **Identifiers** tab.
4. Set **mu = 0** and press **Enter**. So we have just turned off damping.

5. Click the **Integration** button.

It will take you some seconds to finish the simulation. In the Fig. 2.10 you can see the graphical window after two numerical experiments.

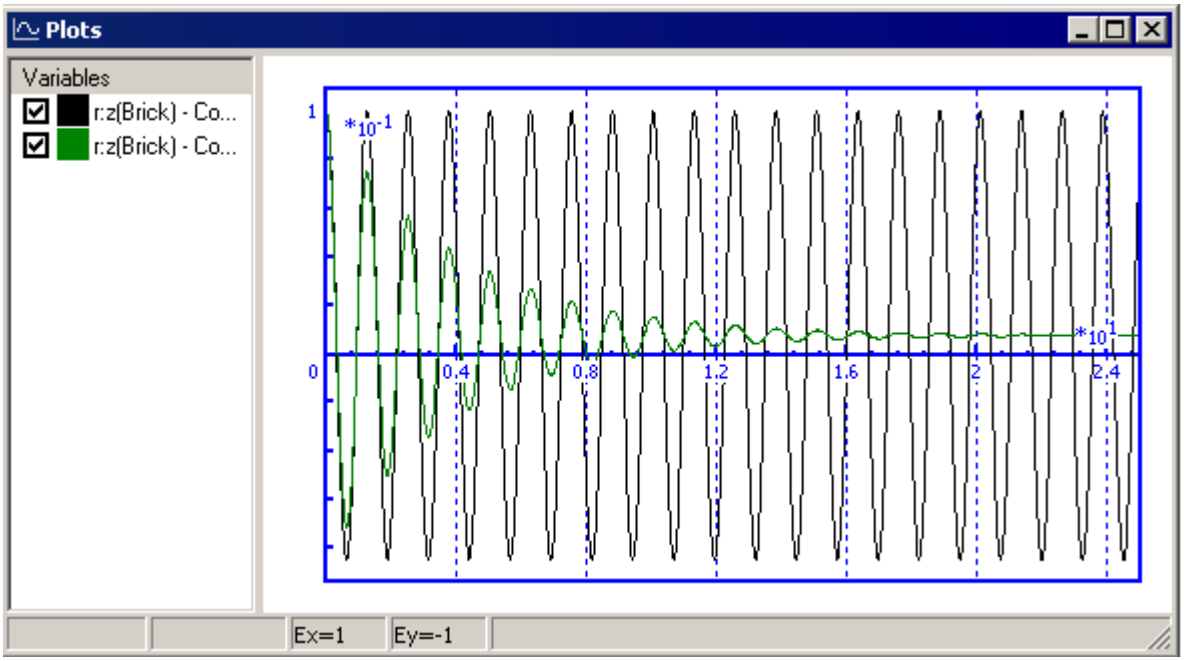


Figure 2.10. Graphical windows after the first experiment

Free oscillation: critical damping

As we showed above critical damping coefficient is **mu = 100** Ns/m. Let us analyze the motion of the **Brick** in such a case.

1. Point to the graphical windows. Select the first variable **r:z(Brick)** and copy it as a static one again (use the **Copy as static variable** item from the context menu**)**.
2. Select the **Pause** inspector and click the **Interrupt** button. **Object simulation inspector** appears.
3. Select the **Identifiers** tab and set **mu = 100**.
4. Click **Integration**.

Now you can see that the motion of the brick is non-periodic, see the Fig. 2.11.

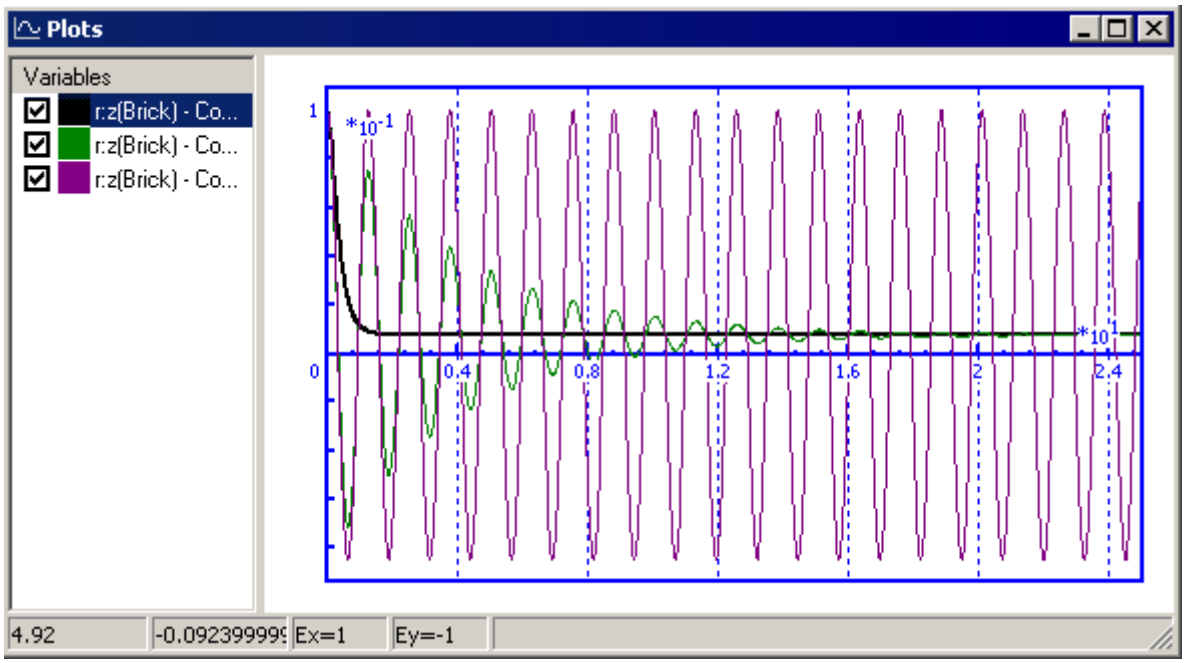


Figure 2.11. Graphical window after three numerical experiments

5. Make numerical experiments for other values of the damping coefficient. Do not forget to copy variables as static ones.
6. If you changed the value of the damping coefficient, set it again to **mu = 100** Ns/m.

2.4.2. Statistical analysis

Now we will come through some additional tools for analysis of results of the simulation.

1. From the **Tools** menu select **Statistics.** New **Statistics** window appears.
2. Drag the variable, which corresponds to free oscillations, from the graphical window to the **Statistics** window.
3. Select the **Statistics** window and point to **Power spectral density**.

The characteristic shape of the power spectral density shows the process has the only frequency, which corresponds to natural frequency. We have the accurate analytical solution – 5 rad/s. Not let us obtain this frequency numerically from the plot of the power spectral density, see the Fig. 2.12. It is approximately **0.78** Hz, see abscissa in the left bottom corner, 0.78 Hz gives us $0.78 \cdot 2\pi = 4.9$ rad/s. You can see that numerically obtained values are quite close to analytical one.

Note. To pick the frequency more precisely use changing scale of the window as it is shown in the Fig. 2.12.

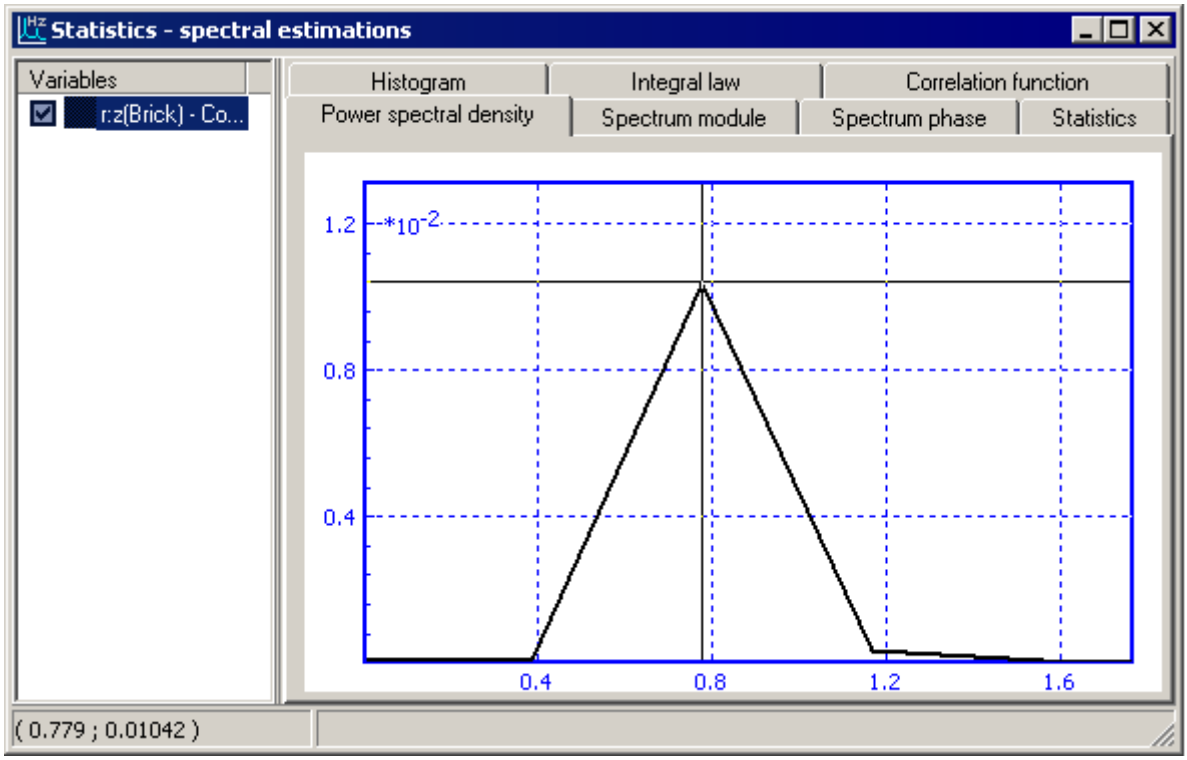


Figure 2.12. Power spectral density of the free oscillations

2.4.3. Linear analysis

Let us consider an example of using the **Linear analysis**. With the help of this tool we will find the equilibrium position of the system, its natural frequencies and forms, define how much the actual damping ration relative to the critical one.

Well, at first you need to close **Pause** and **Object simulation inspector** windows.

1. Select the **Pause** windows and click **Interrupt**.
2. Select the **Object simulation inspector** and click **Close**.

Open Linear analysis window

3. From the **Analysis** menu select **Linear analysis**. The **Linear analysis** window appears.

Equilibrium position

4. In the **Linear analysis** window select the **Equilibrium** tab. Click the **Compute** button. You can see the message "**Equilibrium position is successfully computed!**". The brick in the animation window is now in its equilibrium position.

Note. Obtained coordinates, which correspond to equilibrium position, you can save to a file. To do it use the 🖫 button in the **Initial conditions** tab. This file with initial conditions you can loaded using **Object simulation inspector** in order to start simulation form the equilibrium position if necessary.

Natural frequencies and forms

5. Select the **Frequencies** tab. In the left list you can see the natural frequencies of the system. As you can see our system has only one frequency and this frequency is 0.795775 Hz, what corresponds to 5.0000 rad/s.
6. Click the **Show** button to start animation of the natural forms. Adjust appropriate **Amplitude** and **Rate**. Click the **Stop** button to finish animation.

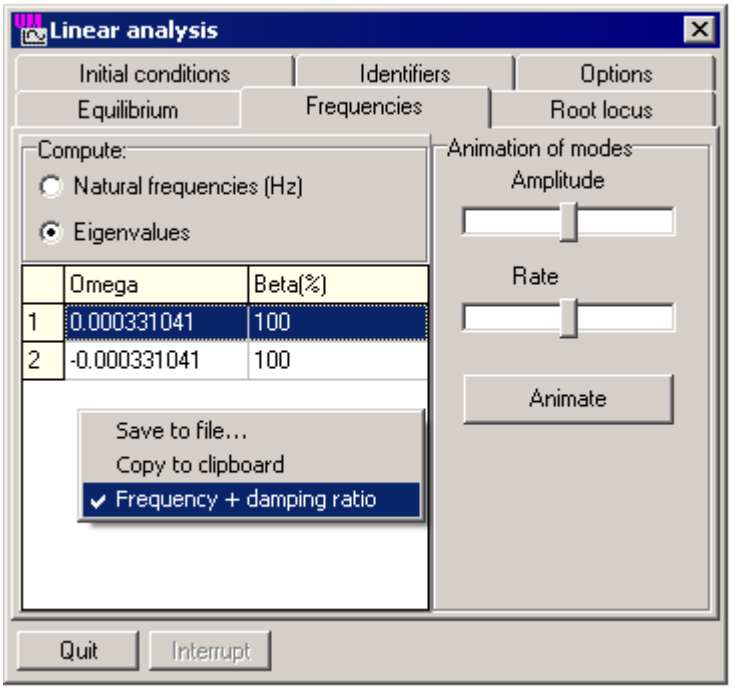
Stability

Let us find the roots of the linearized system. It gives us the information about stability of the model.

7. Set the **Compute** to **Eigenvalues**. You can see that the real parts of roots are negative, therefore system is stable.

Damping ratio

8. Let us describe the damping ratio of the system. Click the right mouse button on the list of eigenvalues and from the context menu select the **Frequency + damping ratio** menu command. We have **Beta = 100 %**, that corresponds to critical damping.



Note. Damping ration shows us if all forms are damped properly and thus change damping coefficients or geometry of attachment point of dampers if necessary.

9. You can change the value of the **mu** identifier in the **Identifiers** tab and see what will happen with damping ratio.
10. Close the **Linear analysis** window.

2.4.4. Forced oscillations

Let us consider simulation of forced oscillations without damping.

1. Delete all variables from the graphical window except the first (dynamic) one.
2. From the **Analysis** menu select **Simulation**.
3. Select the **Identifiers** tab. Set the following values: **a = 0.05**, **omega = 8**, **mu = 0**.
4. Run integration. Now you can see that the body **Top** also moves. The time history of the vertical position of the **Brick** is given in the Fig. 2.13.

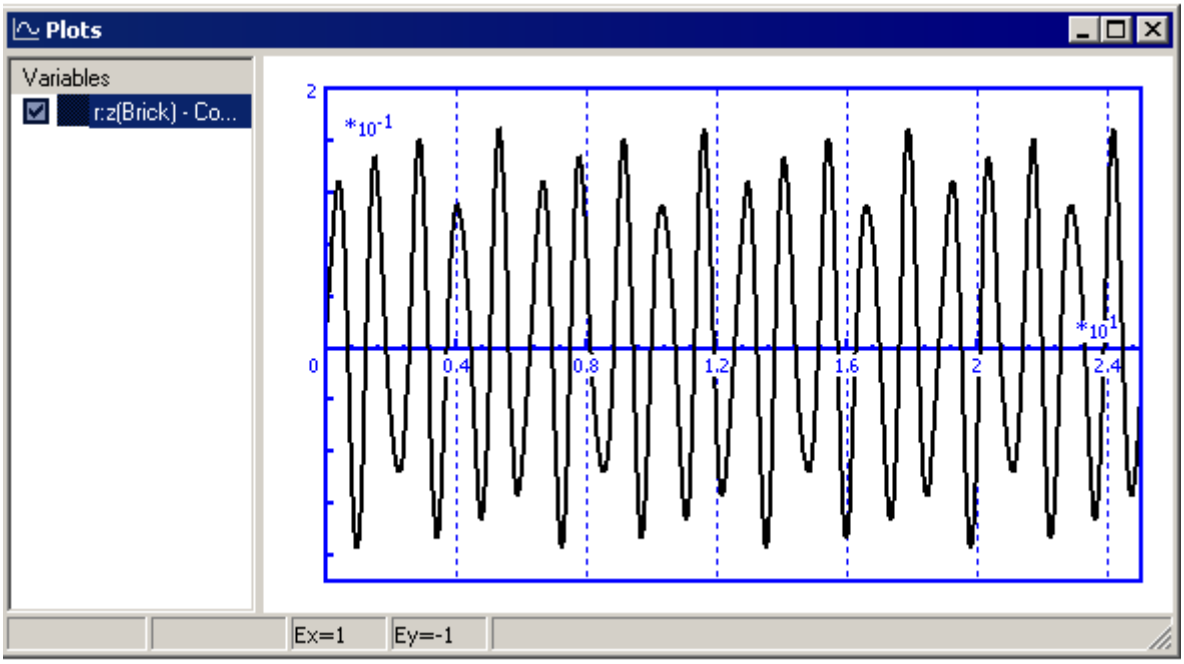


Figure 2.13. Forced oscillation (omega = 8 rad/s)

Resonance

In conclusion we consider the resonance case, when the excitation frequency is equal to the natural frequency of the system.

1. In the **Pause** window click the **Interrupt** button.
2. In the **Object simulation inspector** set **omega = 5**.
3. Run integration. As we expected in the resonance case the amplitude of the oscillations increases in the long run, see the Fig. 2.14.

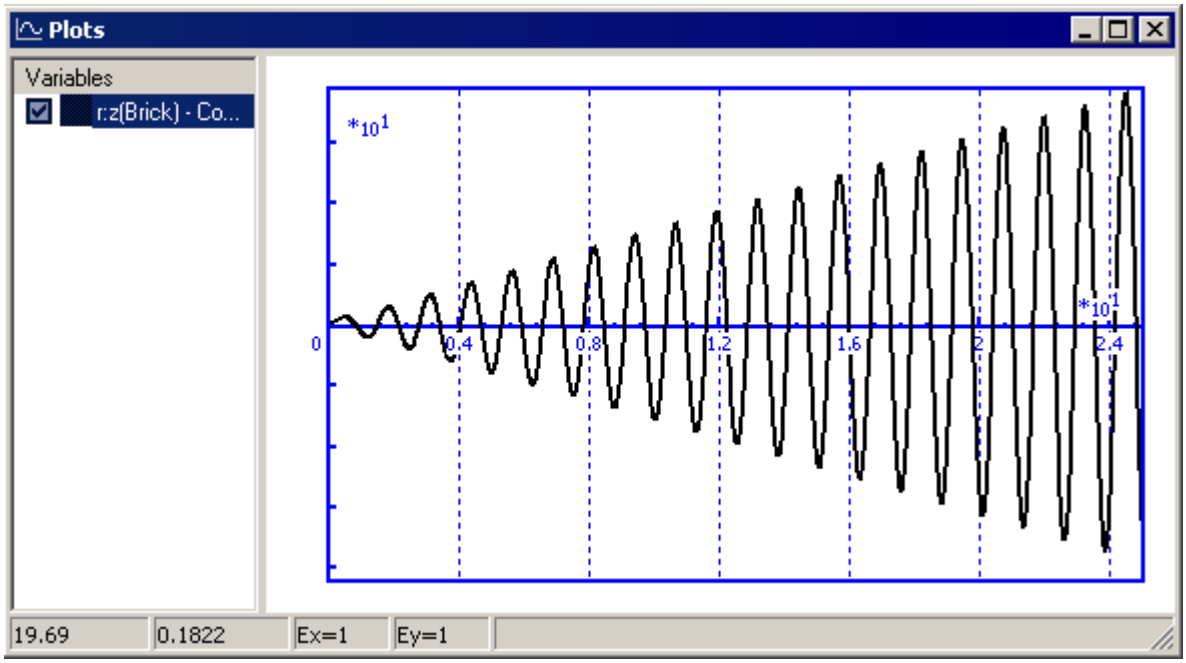


Figure 2.14. Forced oscillations: resonance case

3. Subsequent studying Universal Mechanism

You have come through two examples of dynamical systems (pendulum and sprung body) and have seen the basic tools and features of the UM Base version.

The *Getting Started* series includes other manuals that devoted to the rest modules of the Universal Mechanism. Here they are:

- Getting Started: simulation of road vehicles;
- Getting Started: railway vehicle dynamics;
- Getting Started: Matlab/Simulink interface;
- Getting Started: scanning and optimization module;
- Getting Started: simulation of flexible bodies with UM FEM;
- Getting Started: durability analysis.

Library of simple models: how to…

The UM User's Manual includes **07_UM_Simulation_Examples.pdf**, which is devoted to consideration of simple models that show you how to create/model various graphical elements, joints and force elements. Studying these examples helps you familiarize yourself with basics of Universal Mechanism and approaches for simulation of objects of different kind. The library of models is in the **{um_root}\samples\library** directory.

The **07_UM_Simulation_Examples.pdf** you can find in the **{um_root}\manual** directory or download using the following link: http://www.umlab.ru/download/50/eng/07_um_simulation_examples.pdf.